**无人机遂行编队飞行中的纯方位无源定位设计**

摘要

本文主要研究无人机纯方位无源定位,建立定位模型和调整方案模型完成有效定位被动接受信号无人机以及进行纯方位无源定位调整工作.

针对问题一(1),使用了**余弦定理**和**遍历算法** ,首先确定信号接收点的偏差范围,然后建立以FY00为圆心,R为半径的极坐标系.之后确立信号发射点FY0i,FY0j,因为发射点为理想坐标,所以可以使用极坐标来表示两点方位,然后设接收点FY0m坐标为,由此在表示出来所有点的坐标后,便可使用**余弦定理**建立方位坐标与方向信息之间的关系表达式,然后将已知的方向信息代入关系表达式中,即可反推出FY0m的实际坐标.最后使用**迭代算法**,在固定一个理想点作为发射无人机的发射点,遍历圆周上所有的点,观察其实际坐标在误差范围内,模型合理.

针对问题一(2),使用了**三点定位法**和**遍历算法，**得出还需要一架无人机作为信号发射机就可以完成定位。我们的已知信息是当不共线的三个点，组成的三个圆形可以确定一个定。回到本题中，可以用这个模型来完成。首先确定两个发射无人机的位置，然后通过遍历的方法，将第三个圆的每种情况都考虑出来，然后通过**模拟仿真算法，**对结果修正，验证模型正确性，即证。

针对问题一(3),我们的已知信息是发射机编号已知,位置略有偏差,接收机位置存在偏差,调整是只能使用方向信息.所以我们先规定**坐标的误差范围**以及**方向信息的误差范围,**然后由题目信息可知FY00与FY01为标准坐标,所以通过计算每个点实际位置坐标与理想位置坐标的**偏差程度,**然后选出误差最小的点(在FY02附件)作为另一个发射点.然后根据问题一中**三点即可定位**的条件以及**余弦定理**的公式推导,将FY00,FY01, 作为发射点,**遍历**剩余所有实际位置,选择一个坐标误差与方向信息误差同时满足最小条件另一个点FY0i,然后通过**调整方向信息**,将FY0i调整到最优位置,然后将它的误差值与做对比,选择一个误差值最小的作为新的发射点,然后遍历除 或FY0i以外的剩余点,重复上述工作,直到所有点都在可接受误差范围以内时,位置调整完成,迭代结束.

# 问题重述

## 问题背景

无人机集群在执行编队飞行任务时，在保持电磁静默的情况下，避免外界干扰并维持编队的队形。为了达到这个目的，提出了一种基于纯方位无源定位的方法来调整无人机的位置。具体而言，该方法通过编队中的某些无人机主动发射信号，而其余无人机则被动接收这些信号。接收的无人机通过提取方向信息来进行定位，并据此调整自己的位置。

在这种方法中，编队中的每架无人机都拥有一个固定编号，并且与其他无人机之间的相对位置关系保持不变。接收信号的无人机会接收到两个发射信号无人机之间的夹角方向信息，并根据这一信息来定位和调整自己的飞行位置，以确保整个编队的队形保持稳定和准确。这种方法有效减少了电磁波的外部发射，从而降低了被外界干扰和探测的风险，提高了无人机编队飞行的隐蔽性和安全性。

## 问题提出

编队由10架无人机组成，形成圆形编队，其中 9 架无人机（编号 FY01~FY09）均 匀分布在某一圆周上，另1架无人机（编号 FY00）位于圆心（见图 2）。无人机基于自身感知的高度信息，均保持在同一个高度上飞行。通过建立数学模型，解决下列问题。

问题一：

（1）位于圆心的无人机（FY00）和编队中另 2 架无人机发射信号，其余位置略有偏差的无人机被动接收信号。当发射信号的无人机位置无偏差且编号已知时，建立被动接收信号无人机 的定位模型。

（2）某位置略有偏差的无人机接收到编号为 FY00 和 FY01 的无人机发射的信号，另接收到编队中若干编号未知的无人机发射的信号。若发射信号的无人机位置无偏差，除 FY00 和 FY01 外，还需要几架无人机发射信号，才能实现无人机的有效定位？

（3）) 按编队要求，1 架无人机位于圆心，另 9 架无人机均匀分布在半径为 100 m 的圆周上。当初始时刻无人机的位置略有偏差时，请给出合理的无人机位置调整方案，即通过多次调整， 每次选择编号为 FY00 的无人机和圆周上最多 3 架无人机遂行发射信号，其余无人机根据接收到的方向信息，调整到理想位置（每次调整的时间忽略不计），使得 9 架无人机最终均匀分布在某个圆周上。利用表格给出的数据，仅根据接收到的方向信息来调整无人机的位置，请给出具体的调整方案。

问题二：

实际飞行中，无人机集群也可以是其他编队队形，例如锥形编队队形仍考虑纯方位无源定位的情形，设计无人机位置调整方案。

# 问题分析

## 对问题一的分析

### 第一小问的分析

针对第一小问，使用了**三角函数模型和遍历算法**。因为我们需要建立被动接收信号无人机的定位模型，其本质目标是在确定好信号发射源位置之后，**求出接受无人机的具体坐标**，并观察其与实际坐标之间的误差，**确定模型的合理性。**

以中心位置无人机为原点，建立以R为半径的极坐标。首先固定一个点作为信号发射源，让另一个信号发射源持续移动，观察三者之间的角度关系，然后以此分类。在每个分类中，可以根据相应角度关系，边长关系，余弦定理，勾股定理，由此表示出接收点的坐标（一个极坐标方程）。

### 第二小问的分析

针对第二小问，首先根据题目要求我们确定编号为FY00和FY01为发射点，还需要确定一个未知的发射点，和存在位置偏差的的信号接收机。

我们已知的条件为发射信号的两个无人机的坐标分别为（0，0），（R，0），假设还需要一个编号为i的无人机发射信号，该无人机的坐标为，当我们假设出该接收点相对于中心发射点的偏离角度,使用**三点定位法**，首先模拟出该接收点在圆上的理想位置,画出以三个发射点为圆心，发射点到接收点为半径的三个圆，由图4可知，三个圆的交点为信号接收点。但因为信号接收点出现位置偏差，所以实际情况应该为图5所示，为判断信号接收点的具体位置，我们将三个圆两两进行分组，分别求出三个点为信号接收点，之后取三个点的坐标的均值作为终端的坐标,这样即可确定出该偏离点的具体位置。之后使用**遍历算法,**遍历在该偏差下i取不同值的位置坐标,验证模型是否具有普遍性和合理性。

### 第三小问的分析

针对第三小问，首先通过题目已知发射机可以指定，即发射机编号已知，位置略有偏差，接收点位置略有偏差，调整时只能使用方向信息去调整接收机位置角度。由问题（1）可知，圆心+两个圆周上发射点可定位，所以我们需要确定出三个发射点的坐标位置，然后才能推算出其余各点的坐标。

第一步，我们选择半径为100的基本圆，设定误差范围为（95R105）,（）由题目可知，FY00与FY01为标准发射点，所以我们需要在确定一个在误差范围之内，偏离程度最小的无人机作为发射点，然后遍历其余所有点作为接收机，使用问题（1）中的方向信息的计算方法，观察在该误差下，哪一个点的实际方向信息与理想方向信息差距最小，然后通过移动该点的位置，使它无限靠近理想坐标位置，然后以该点与FY00与FY01作为新的发射点，重新遍历剩余所有接收无人机，重复迭代，直到所有无人机坐标都在误差范围之内，循环结束，由此，所有无人机位置确定。

## 对问题二的分析

# 模型假设

1、无人机在无风、无障碍的理想环境中飞行。

2、无人机的飞行高度恒定，且均保持在同一水平面上。

3、无人机信号传播不受干扰和衰减，信号传输速度恒定。

4、每架无人机都具有唯一编号，并且编号在整个飞行过程中保持不变。

5、编队中的每架无人机能够精准测量与其他无人机的相对位置。

6、发射信号的无人机位置无偏差，接收信号的无人机位置可能存在微小偏差。

7、每架无人机能够接收并处理来自多个方向的信号，并计算夹角方向信息。

8、初始时刻，编队中位于圆心的无人机位置准确，圆周上的无人机位置可能略有偏差。

# 符号说明

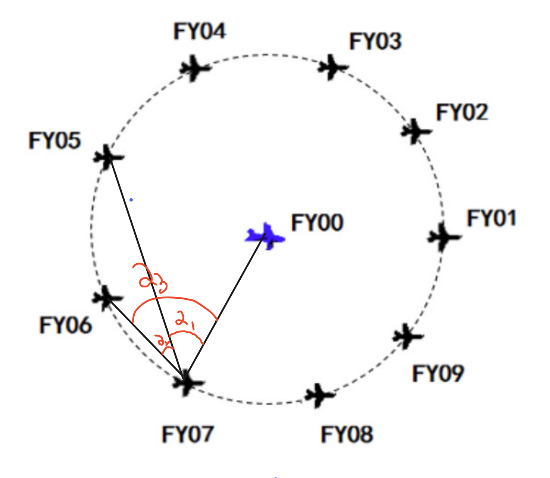
|  |  |
| --- | --- |
| **符号** | **说明** |
|  | FY0i和FY00与FY0m所形成的夹角. |
|  | FY0i和 FY0j与FY0m所形成的夹角. |
|  | FY00和FY0j与FY0m所形成的夹角. |
| a | x1-x2 |
| b | y1-y2 |
| c | x1-x3 |
| d | y1-y3 |
| e |  |
| f |  |

# 模型的建立与求解

## 问题一的模型建立与求解

### 第一小问的模型建立与求解

该问题使用了**三角函数模型和遍历算法**。由题可知，中心点FY00作为已知发射点，还需要确定另外两个发射点，所以由于各无人机在圆周上均匀分布，由图可知，我们可以假设FY05 为另一个固定发射点，遍历圆周上所有点作为最后一个发射点，观察三个点之间的相对位置, FY05与接收点所连线段与FY00与接收点所连线段形成夹角为;FY05与接收点所连线段与FY0i与接收点所连线段形成夹角为；FY00与接收点所连线段与FY0i与接收点所连线段形成夹角为（角度表示如图一所示）

图1 角度表示图（以FY0i = FY06为例）

建立以FY00为圆心，半径为R的极坐标坐标系。

1. 圆周上编号为FY01到FY09的无人机的位置可以通过极坐标表示为：

, , k = 1, 2, …,9

1. 将极坐标转化为直角坐标：





1. 假设FY00，FY0i，FY0j三架无人机发射信号，我们需要编号为FY0m(mi,m无人机接收到的方向信息，这些方向信息为该无人机与发射信号的三架无人机连线之间的夹角。
2. 编号为FY0m的无人机接收到的方向信息为夹角，，，我们可以使用余弦定理来计算这些角度。

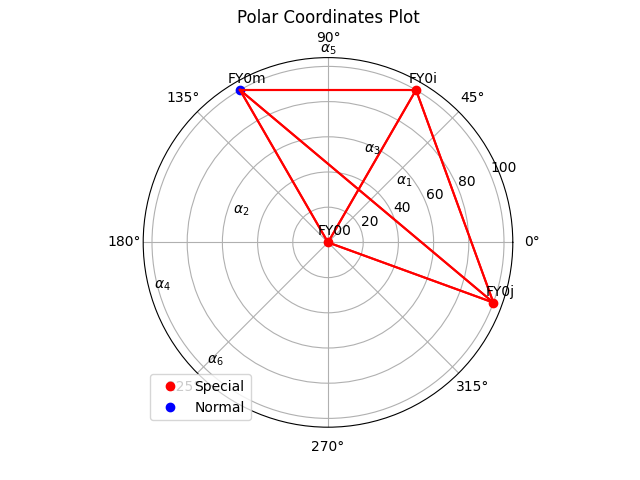
假设被动接收信号的无人机FY0m的位置为（,）, 它与发射信号的FY00，FY0i和FY0j无人机关系的连线如图所示：



 图2 边和角之间的关系

1. 通过方向信息进行定位,使用余弦定理求解夹角.(为FY0i在极坐标中的角度,)

如图所示:FY00位于圆心:(0,0)

FY0i位于

FY0j位于

被接受信号的无人机编号为FY0m,其坐标为 我们需要计算的夹角是（见符号说明）计算各边的长度:







=



1. 计算方向信息:





1. 利用方向信息进行定位

通过已知的，以及发射信号的无人机的坐标（0，0）、和，我们可以通过遍历算法，遍历每个无人机在它的误差范围内所对应的发射点的位置，反解求出被动接收信号的无人机FY0m的坐标。

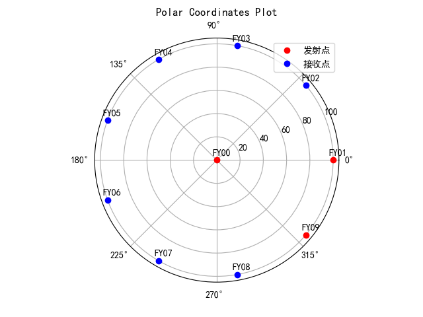
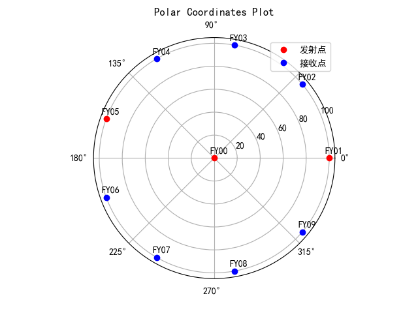
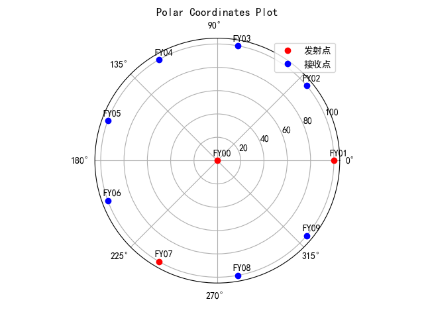
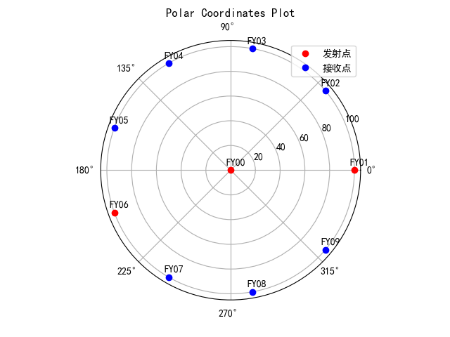
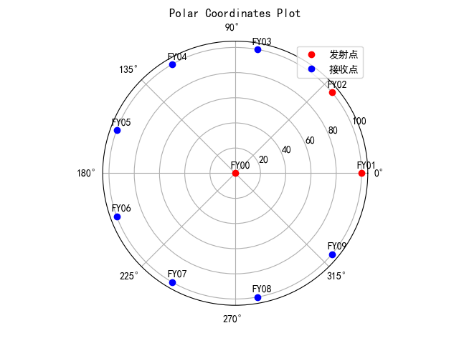
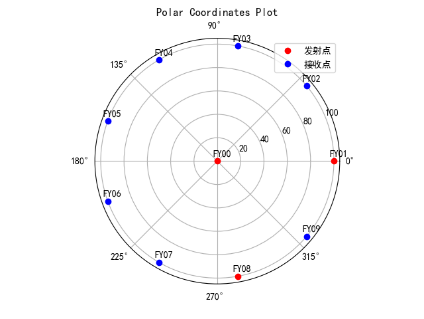
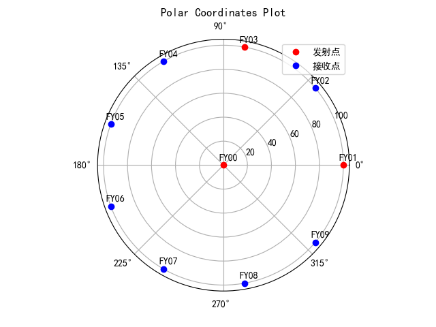
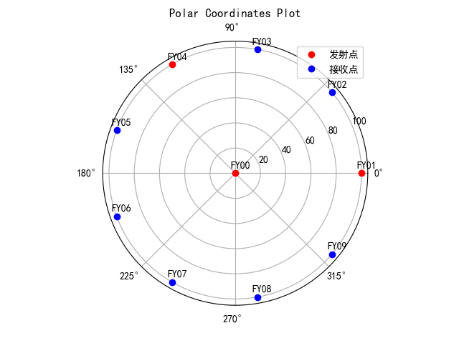
1. 问题一（1）的结果分析遍历结果如图所示：

图3 定位结果展示图 ­­­­­­­­

由图可知，三个红色的点为发射信号点，蓝色的点为信号接收点，我们确定其中一个发射点为（R，0），蓝色点群是被动接收无人机m的实际坐标可能位置。如图可以看出被动接收无人机的实际位置并没有完全分布在圆周上，但误差均在可接受范围内。模型具有准确性和合理性。

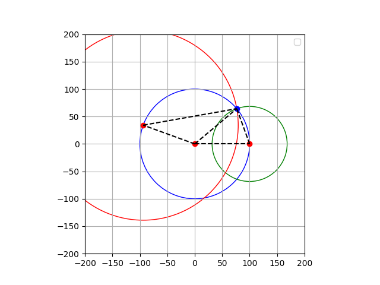
-

### 第二小问的模型建立与求解

1. 模型的建立：**三点定位法,遍历算法**

以三个点作为发射点，去确定一个接收点的位置，通过测量接收点与发射点之间的距离，以该距离作为半径，发射点作为圆心，最终求得三圆的交点，交点即为接收点的位置，达到定位效果。

1. 模型的求解：

：假设理想接收点的位置为FY02，则通过计算出三圆的交点,确定接收点的位置就在圆上(三圆的交点)，由此便可确定接收点的理想坐标,达到定位效果。如下

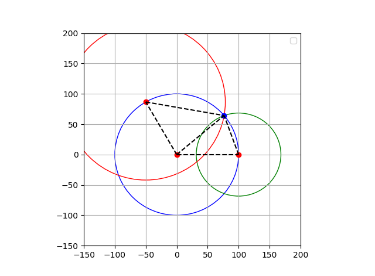
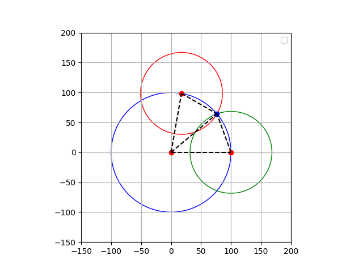
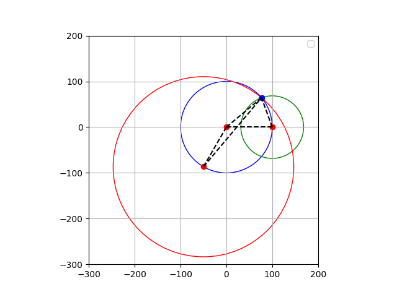
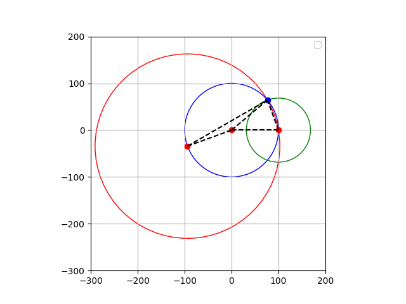
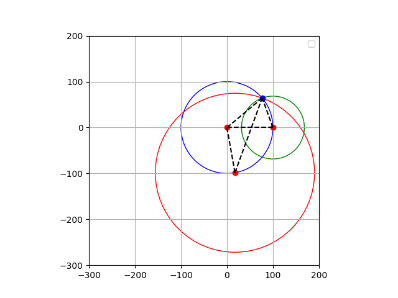
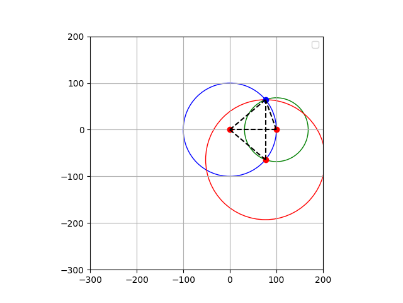
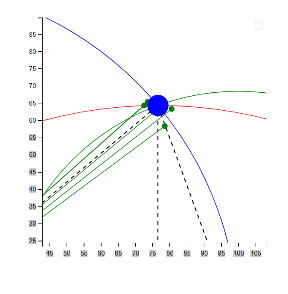
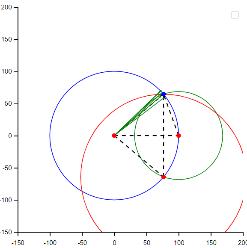
图所示：（红色为发射点，蓝色为接收点）

图4 理想接收点位置图

：但在实际问题中，由于接收点位置存在偏差，所以该点并不在圆上，所以由该点所确定的三个圆之间两两相交形成三个交点，三个交点所形成的面为该接收点的误差范围（如图五所示），然后分别对三个点的坐标均值作为终端的坐标



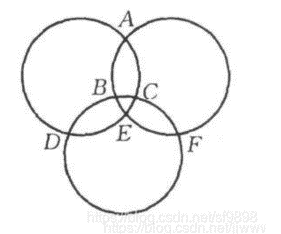
图5 实际接收点位置分布图

图6 实际接收点概念图

：在一个圆上随便取三个点坐标分别为那么有

****





图7 随机生成点定位

:假定情况如图4第一个图所示，随机生成除两个发射点外其他接收点生成一千个角度偏差为1度的随机生成点，利用**第一小问计算方向信息公式**对随机点进行定位处理，将修正的信息与标准结果做比较，得出再需要一架无人机就可定位出轨道有偏差无人机的位置

第一种情况：

表1 偏离角度表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ∠1 | ∠2 | ∠3 | ∠1偏离值 | ∠2偏离值 | ∠3偏离值 | 修正后极坐标 |
| 67.65 | 69.34 | 136.99 | -2.35 | -0.66 | -3.01 | 40 |
| 68.93 | 69.73 | 138.66 | -1.07 | -0.27 | -1.34 | 40 |
| 69.84 | 70.07 | 139.91 | -0.16 | 0.07 | -0.09 | 40 |
| 73.30 | 70.91 | 144.21 | 3.30 | 0.91 | 4.21 | 40 |
| 70.00 | 70.00 | 140.00 | 0.00 | 0.00 | 0.00 | 40 |
| 70.26 | 70.13 | 140.39 | 0.26 | 0.13 | 0.39 | 40 |
| 73.34 | 72.40 | 145.74 | 3.34 | 2.40 | 5.74 | 40 |
| 68.12 | 66.97 | 135.09 | -1.88 | -3.03 | -4.91 | 40 |
| 67.32 | 69.74 | 137.06 | -2.68 | -0.26 | -2.94 | 40 |
| 70.01 | 70.00 | 140.01 | 0.01 | 0.00 | 0.01 | 40 |

剩余表格见附录

据表格可知每次修正完的接收点与标准值相等，且每个位置的修正率达到100%由此可知，还需要一架无人机就可实现精准定位。

### 第三小问的模型建立与求解

**模型的建立：**

Step1:确定坐标误差范围为（95R105）,（）,方向信息误差范围为

Step2:确定除FY00,FY01外最小误差点作为信号发射机

设理想状况下无人机相对圆心的角度为（i = 1,2,3,…,9），现实情况下无人机相对圆心的角度为（i = 1,2,3,…,9）。设理想状况下无人机的相对半径为R（R=100），现实情况下无人机的相对半径为。则理想状况下无人机的坐标为（i = 1,2,3,…,9）,现实状况下无人机的坐标为（i=1，2，3，…,9）。

所以实际状况与理想状况下的相对误差为（i=1，2，3…，9）。遍历除FY01外所有点，经计算可得，FY02为偏离误差最小的点。

Step3: 当FY00，FY01，FY02为发射点时，遍历圆上另外所有点作为接收点，利用问题（1）中的余弦定理的方法与模型，可以求出来每一个实际接收点的方向信息。比较每个接收点理想方向信息与之间的误差大小，然后选择方向信息误差最小的点FY0i，然后调整FY0i的使它既满足在理想位置的误差范围之内，又满足坐标在方向信息的误差范围之内，然后经过多次调整，得到最优解。

Step4:在确定好FY0i的最优位置后,比较FY0i与FY02的坐标的相对误差的大小,如果FY0i的坐标的相对误差比FY02的小,则选择FY0i与FY00,FY01重新作为发射点,重新遍历除FY02以外的所有接收点;反之,选择FY02,FY00,FY01重新作为发射点,重新遍历除FY0i以外的所有接收点.

Step5:重复迭代上述过程,直到所有点都在规定误差范围之内,则模型合理.

Step6:最终结果展示:

|  |  |  |  |
| --- | --- | --- | --- |
| **无人机名称** | **实际极坐标** | **标准极坐标** | **差值(取范数)** |
| FY00 | (0, 0) | (0, 0) | 0 |
| FY01 | (100, 0) | (100, 0) | 0 |
| FY02 | (98, 40.1) | (100, 40) | 2.007449252 |
| FY03 | (112, 80.21) | (100, 80) | 12.00626738 |
| FY04 | (105, 119.75) | (100, 120) | 5.019950683 |
| FY05 | (98, 159.86) | (100, 160) | 2.014574616 |
| FY06 | (112, 199.96) | (100, 200) | 12.00022745 |
| FY07 | (105, 240.07) | (100, 240) | 5.001567011 |
| FY08 | (98, 280.17) | (100, 280) | 2.021453359 |

|  |  |  |  |
| --- | --- | --- | --- |
| **无人机编号** | **实际方位角度** | **理想方位角度** | **角度差值** |
| 3 | (46.05, 61.1, 15.05) | (50.0, 70.0, 20.0) | (3.95, 8.9, 4.95) |
| 4 | (29.31, 48.45, 19.14) | (30.0, 50.0, 20.0) | (0.69, 1.55, 0.86) |
| 5 | (10.17, 30.41, 20.23) | (10.0, 30.0, 20.0) | (0.17, 0.41, 0.23) |
| 6 | (9.41, 9.45, 18.86) | (10.0, 10.0, 20.0) | (0.59, 0.55, 1.14) |
| 7 | (29.23, 9.79, 19.44) | (30.0, 10.0, 20.0) | (0.77, 0.21, 0.56) |
| 8 | (50.78, 30.42, 20.36) | (50.0, 30.0, 20.0) | (0.78, 0.42, 0.36) |
| 9 | (61.23, 46.26, 14.97) | (70.0, 50.0, 20.0) | (8.77, 3.74, 5.03) |

由第一张表可知，FY02无人机的偏差最小，应选择为理想状态下无人机，由第二张表格可知，FY02的角度偏差最小，此时再和FY02无人机的角度偏差作比较，然后依次迭代求出最优解。最终得到的修正结果如下表所示

|  |  |
| --- | --- |
| **无人机名称** | **实际极坐标** |
| FY00 | (0, 0) |
| FY01 | (100, 0) |
| FY02 | (99.52, 40.05) |
| FY03 | (100.21, 80.21) |
| FY04 | (101.21, 119.75) |
| FY05 | (99.50, 160.86) |
| FY06 | (100.10, 199.96) |
| FY07 | (100.80, 240.07) |
| FY08 | (100.20, 280.17) |

## 问题二的模型建立与求解

# 参考文献

# 附录

|  |
| --- |
| 附录1 |
| 介绍：支撑材料的文件列表 |
|  |

|  |
| --- |
| 附录2 |
| 介绍：该代码是某某语言编写的，作用是什么 |
| import numpy as np  import itertools, random  from 极坐标绘制 import Draw  import matplotlib.pyplot as plt  import math  print(math.cos((60/180)\*math.pi))  class Points:  def \_\_init\_\_(self, r, d, name=''):  self.radius = r  self.d = d  self.is\_shoot = False  self.signal = []  self.polar\_numpy\_list = np.array([self.radius, self.d])  self.right\_angle = self.polar\_to\_cartesian(self.radius, self.d)  self.special = False  self.edge = []  self.name = name  def polar\_to\_cartesian(self, r, theta):  theta = np.radians(theta)  x = r \* np.cos(theta)  y = r \* np.sin(theta)  return np.array([x, y])  def \_\_str\_\_(self, \*args, \*\*kwargs):  return (  f'name is {self.name}\n'  f'self.numpy\_list is {self.polar\_numpy\_list}\n'  f'self.edge is {self.edge[0:10]}\n'  f'self.signal is {self.signal[0:10]}\n'  f'self.is\_shoot is {self.is\_shoot}\n\n\n'  )  class Coordinates(object):  def \_\_init\_\_(self,name):  self.name = name  self.ob\_ls = []  self.all\_ls = []  self.shoot\_ls = []  self.draw = Draw  def running(self, show=False):  self.shoot()  self.calculate\_distance()  self.calculate\_signal()  if show:  print('\n' \* 3, ''"经过计算完成后的", '\n' \* 3)  self.show\_shoot()  def shoot(self, bound=(0, 0), number=1):  # for item in self.shoot\_ls:  # item.is\_shoot = True  for item in self.all\_ls:  if item.is\_shoot and item not in self.shoot\_ls:  self.shoot\_ls.append(item)  if not item.is\_shoot and item not in self.ob\_ls and not item.special:  # print('进入')  self.ob\_ls.append(item)  if item in self.shoot\_ls and item in self.ob\_ls:  self.ob\_ls.remove(item)  # for item in self.all\_ls:  # if np.all(item.polar\_numpy\_list) in bound:  # self.shoot\_ls.append(item)  # item.is\_shoot = True  # while True:  # c = random.sample(self.all\_ls, 1)  # print(c[0])  # if len(self.shoot\_ls) >= number:  # break  # if c[0] not in self.shoot\_ls:  # c[0].is\_shoot = True  # self.shoot\_ls.append(c[0])  # self.ob\_ls = [item for item in self.all\_ls if item not in self.shoot\_ls]  def calculate\_distance(self):  for shoot\_signal in self.shoot\_ls:  coords = shoot\_signal.right\_angle  for receive\_signal in self.ob\_ls:  rev = receive\_signal.right\_angle  edge = rev - coords  if np.all(edge[1] != 0):  # print('edge is ', edge)  shoot\_signal.edge.append(edge)  receive\_signal.edge.append(edge)  else:  print("error")  def calculate\_signal(self):  def angle\_between\_vectors(v1, v2):  dot\_product = np.dot(v1, v2)  norms = np.linalg.norm(v1) \* np.linalg.norm(v2)  cos\_theta = dot\_product / norms  angle = np.arccos(cos\_theta)  return np.degrees(angle)  for item in self.ob\_ls:  polar\_coords = item.edge  print(polar\_coords)  combinations = list(itertools.combinations(polar\_coords, 2))  for item\_0 in combinations:  item.signal.append(angle\_between\_vectors(item\_0[0], item\_0[1]))  # item.signal.append(angle\_between\_vectors())  def vector\_to\_angle(self, vector):  # 确定初始点和向量的参数  P1 = np.array([0, 0]) # 初始点P1  length\_A = 100 # 向量A的长度  length\_B = 100 # 向量B的长度  angle\_between\_vectors = 40 # 向量A和B的夹角（度数）  # 确定向量A的角度（假设为0度）  alpha = 0 # 向量A相对于x轴的角度（度数）  # 计算P2的坐标  x2 = P1[0] + length\_A \* np.cos(np.deg2rad(alpha))  y2 = P1[1] + length\_A \* np.sin(np.deg2rad(alpha))  P2 = np.array([x2, y2])  # 确定向量B的角度  beta = alpha + angle\_between\_vectors  # 计算P3的坐标  x3 = P2[0] + length\_B \* np.cos(np.deg2rad(beta))  y3 = P2[1] + length\_B \* np.sin(np.deg2rad(beta))  P3 = np.array([x3, y3])  print(f'P1的坐标: {P1}')  print(f'P2的坐标: {P2}')  print(f'P3的坐标: {P3}')  def show\_shoot(self):  print(f"{self.name} is showing")  print('\n' \* 5, '\*' \* 10, '\n' \* 5, 'SHOW\_ALL')  for item in self.all\_ls:  print(item)  print('\n' \* 5, '\*' \* 10, '\n' \* 5, 'SHOW\_OB\_LS')  for item in self.ob\_ls:  print(item)  print('\n' \* 5, '\*' \* 10, '\n' \* 5, 'SHOW\_SHOOT\_LS')  for item in self.shoot\_ls:  print(item)  class Observation:  def \_\_init\_\_(self, radius=100):  self.radius = radius  self.std\_coordinates = [(self.radius, item) for item in range(0, 321, 40)]  self.std\_coordinates.insert(0, (0, 0))  self.dev\_points = [self.polar\_to\_cartesian(item[0], item[1]) for item in self.std\_coordinates]  self.dev\_coordinates\_cart = [self.generate\_random\_points(  item, 10000, 5) for item in self.dev\_points]  a = self.std\_coordinates.pop(2)  self.dev\_points\_plor = [self.cartesian\_to\_polar(item[0], item[1]) for item\_0 in item for item in  self.dev\_coordinates\_cart]  self.dev\_points\_plor.append(a)  def cartesian\_to\_polar(self, x, y):  r = np.sqrt(x \*\* 2 + y \*\* 2)  theta = np.arctan2(y, x) # atan2函数可以处理x为零的情况  return (r, np.degrees(theta))  def polar\_to\_cartesian(self, r, theta):  theta = np.radians(theta)  x = r \* np.cos(theta)  y = r \* np.sin(theta)  print('x,y:', x, y)  return np.array([x, y])  def generate\_random\_points(self, center, num\_points, radius):  """  在给定中心点附近生成随机分布的点。  参数:  center (tuple): 中心点的坐标 (x, y)  num\_points (int): 要生成的随机点的数量  radius (float): 随机点分布的半径  返回:  numpy.ndarray: 随机点的数组，形状为 (num\_points, 2)  """  # 生成随机角度  angles = np.random.uniform(0, 2 \* np.pi, num\_points)  # 生成随机半径  radii = np.random.uniform(0, radius, num\_points)  # 计算随机点的坐标  x\_points = center[0] + radii \* np.cos(angles)  y\_points = center[1] + radii \* np.sin(angles)  return np.vstack((x\_points, y\_points)).T  # 示例参数  # 可视化随机点 |

|  |
| --- |
| 附录3 |
| 介绍：该代码是某某语言编写的，作用是什么 |
|  |