无人机遂行编队飞行中的纯方位无源定位问题

摘要

无源定位在无人机遂行编队自动定位中有着重要应用，本文对无人机定位规律以及队列调整方案进行了研究与分析。

针对问题1(1)，分析需要定位的无人机位置与其接收到的方位角之间的关系，并假设需要定位的无人机离理想位置的距离不超过相邻两架无人机理想距离的一半，建立基于**三点定位**的几何模型。首先通过分析无人机编队的几何性质，建立定位无人机与发射信号无人机之间的距离，以及定位无人机接收方位角之间的方程组，并利用**牛顿迭代**求解，得到定位无人机与发射信号无人机之间的距离。然后，利用三点定位法求出定位无人机的准确位置。最后，本文通过实例，对定位方案进行了验证，定位误差平均在0.005%以内。

针对问题1(2)，首先基于问题1(1)中的定位模型，分析出除了$FY00$和$FY01$之外，如果只在圆周上任取一架无人机发射信号，则在位置编号的情形下并不能精确定位，只能得到一系列待定的定位预测点。然后从理论上证明出除了$FY00$和$FY01$之外，另外随机选取两架无偏差的无人机发射信号，即可有效定位，并给出了具体的定位方案。最后，通过实例仿真，验证了定位的可行性，定位误差在0.08%以内。

针对问题1(3)，根据题目中要求所有无人机调整到尽可能均匀分布在某个圆周上的条件，建立以无人机实际位置极径的标准差、实际位置极角同理想极角的标准差为目标的**多目标优化模型**，然后利用贪心的思想搜索合适的发射信号的无人机，并推广问题1(1)中的定位模型，计算出无人机在设定程序下调整到的位置。调整过程中假设无人机可自动将方位角对准至理想方位角同实际接收方位角的加权平均。最后在经过200轮调整后，无人机编队近似地分布在以半径为r的圆周上，其中半径$r \approx 104.85$，各无人机的位置信息详见表 8 和表 9。

针对问题2，在锥形阵列的情形下，基于问题 1(3)中的调整方案设计了无人机在具有微小偏差时的自动调整方案。具体方案为：以锥形阵列中的某点作为原点，将与其相邻的无人机调整到近似位于一个圆周上，且极角差近似为60度。重新选择合适的已调整完的无人机作为坐标原点，重复上述调整方案。以此类推，可将所有无人机排列成锥形，并间距近似相等。最后，通过实例验证了调整方案的可行性，调整后实际位置与理想位置误差平均为$0.0325m$。

关键词：三点定位 牛顿迭代 多目标优化 贪心算法

一、问题重述

1.1 问题背景

无人机发展至今已经经过了几十年，相关技术已经相对成熟,已经在军事和民用领域中发挥重要的作用。为了适应未来的挑战,需要在现有的技术的基础上,

发展更为高效的无人机管理和组织模式。基于此,无人机编队飞行成为提出的无人机合作化发展方向中的一个重要概念。无人机集群在编队飞行时，为避免外界干扰，应尽可能保持电磁静默，少向外发射电磁波信号。为了保持编队队形，拟采用纯方位无源定位的方法调整无人机的位置，即由队伍中某几架无人机发射信号、其余无人机接收信号，从中提取出方向信息进行定位，来调整无人机的位置。

1.2 问题提出

题目约定接收信号的无人机所接收方向信息为：该无人机与任意两架发射信息无人机连线之间的夹角。在此基础上，建立数学模型，解决以下问题。

问题 1：由 10 架无人机组成的圆形编队，一架无人机位于圆心，其余九架均匀分布在某一圆周上，并始终保持在同一高度飞行。

第（1）问：由位于圆心的无人机和另外两架无人机发射信号，其余位置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，建立被动接收信号的无人机定位模型。

第（2）问：某位置略有偏差的无人机除了接收到两架已知编号无人机发射的信号，另接收到若干编号未知的无人机发射的信号。求解还需要最少几架无人机发射信号，才能实现无人机有效定位

第（3）问：九架无人机均匀分布在半径为 100m 圆周上，一架位于圆心，且每次最多 3 架无人机发射信号，利用表格所给数据，根据接收到的方向信息调整无人机位置，使得 9 架无人机最终均匀分布在某个圆周上。

问题 2：考虑其他编队队形，还是遵循纯方位无源定位情形，设计无人机位置调整方案。

二、 问题分析

问题一第（1）小问的分析：题目要求建立被动接收信号无人机的定位模型，其实就是要利用已知信息，寻找几何关系，建立数学模型，从而得到被动接收信号无人机的精确位置。分析发现，能利用三架发射信号无人机的位置以及被动接收信号无人机接收到的三个方向信息，通过三角定位法，精准求出无人机的位置。

问题一第（2）小问的分析：题目要求确立在除去已知的两台发射信号无人机之后，还需要几架无人机，才能实现对其余位置略有偏差的无人机的有效定位。其实就是问至少还需要几架无人机发射信号，才能确定任一被动接收信号无人机的准确位置。因为根据问题一第(1)小问中的定位模型知，必须已知三架发射信号无人机的编号才能准确定位。因此，如果除了 FY00 和 FY01 之外,只增加一架未知编号的无人机发射信号，并不能有效定位。需要额外增加无人机发射信号。本题的关键是在于能否证明或验证，除了 FY00 和 FY01 之外，再增加两架未知编号无人机发射信号，即可有效定位。

问题一第（3）小问的分析：

题目给出处在圆周上的九架无人机的初始位置值，要求给出一个具体调整方案，使得经过调整后，九架无人机能均匀地分布在圆周上。也就是要求建立相应模型，通过求解得到无人机的调整方案。因为题目限制了圆周上最多三架无人机，所以可以引入决策变量，找到约束条件，建立多目标优化模型。设置每次的优化方案，进行多次迭代计算，最终得到每架飞机的具体调

整方案。

问题二的分析：在第二问中，题目要求在无人机的编队队形变为锥形编队队形的情况下，设计无人机位置的调整方案。其实就是要求建立相关数学模型，通过求解得到无人机的位置的调整方案。因为题目给出了无人机是等距分布的，所以可以在上一问的基础上，修改约束条件，建立多目标优化模型。经过多次迭代计算，最终得到锥形编队队形无人机位置的调整方案。

三、 模型假设

假设 1：假设每架无人机有固定的编号，且在编队中与其他无人机相对位置关系保持不变。

假设 2：假设飞机飞行不受天气、障碍物等其他因素的影响。

假设 3：假设无人机都保持在同一个高度上飞行。

假设 4：假设地面只能控制无人机是否发射信号，而不能控制无人机的移动。无人机只能在预先设置好的情况下进行位置调整。

假设 5：被动接收信号的无人机偏差范围较小，超过一定程度的偏差位置不考虑。

假设 6：假设编队中不能有超过四架无人机同时发射信号。

假设 7：假设信号之间互不干扰，无人机角度获取精度高。

四、 符号说明

符号	符号意义
r	圆心到编队中发射信号无人机的距离
d	编队中两架发射信号无人机的距离
x, y, z	第1,2,3架发射信号无人机到定位点距离
n_i	无人机 FY00 到 FY09 的编号，其中$i = 0,1,2,,\ldots,9$
θ_i	第n_i架无人机在位置无偏差时的极角，$i = 1,2,3,\ldots,9$
$\alpha_1, \alpha_2, \alpha_3$	三架无人机发射信号时，被观测无人机所接收到的三个方向信息
$\theta_i{}'$	第n_i架无人机在位置略有偏差时的极角，$i = 1,2,3,\ldots,9$
ρ_i	第n_i架无人机在位置无偏差时的极径，$i = 1,2,3,\ldots,9$
$\rho_i{}'$	第n_i架无人机在位置略有偏差时的极径，$i = 1,2,3,\ldots,9$
δ_i	表示编号为i无人机是否发射信号的决策变量
$Std(\theta_i(\delta))$	编号n_i无人机实际极径与理想极径的标准差

五、模型的建立与求解

5.1 问题一第（1）模型的建立

5.1.1 模型前的准备

第一小问是由位于圆心的无人机和编队中九架无人机的其中两辆发射信号。那么首先我们可以确定发射信号的飞机一共有C_9^2种情况。假设被动接收信号的无人机位置没有偏差，那么，设 FY00，FY01 所在点位置分别为 O,M，以 O 为极点，以 OM 为极径，建立极坐标系。在不考虑偏差时，不妨设所有点极径为单位长度，因为九架无人机是均匀分布的，所以圆周上九架无人机的极角在无偏差时分别如下表所示：

表 1 无偏差时编队无人机的极角

无人机编号	极角(弧度)
n_1	0
n_2	$\frac{2}{9}\pi$
n_3	$\frac{4}{9}\pi$
n_4	$\frac{6}{9}\pi$
n_5	$\frac{8}{9}\pi$
n_6	$\frac{10}{9}\pi$
n_7	$\frac{12}{9}\pi$
n_8	$\frac{14}{9}\pi$
n_9	$\frac{16}{9}\pi$

将编队中的九架无人机从$FY00$开始到$FY09$分别编号为n_i，，其中$i = 1,2,3,\ldots$，9。因为无人机可以通过各种传感器知道自己的高度与水平位置，因此我们可以认为无人机的偏差是较小的。我们假设每架无人机的极径偏差范围为$(0.8r, 1.2r)$，极角的偏差范围为（$\theta_i - \frac{1}{90}\pi, \theta_i + \frac{1}{90}\pi$）。其中，$r$是中心点与发射信号无人机间的距离，$\theta_i$是第$i$架无人机位置没有偏差时的极角，$i = 1,2,\ldots,9$。

5.1.2 第（1）问三角定位模型

首先，我们可以随机确定编队中发射信号的两辆飞机，加上圆心位置发射信号的飞机。在不考虑其余位置的飞机有偏差时，我们能够确定其余七架无人机的具体位置。例如，由 FY01 和 FY02 与 FY00 三架飞机一起发射信号。设圆心为点 O，FY01 与 FY02 所在位置分别为 A、B。如下图所示：

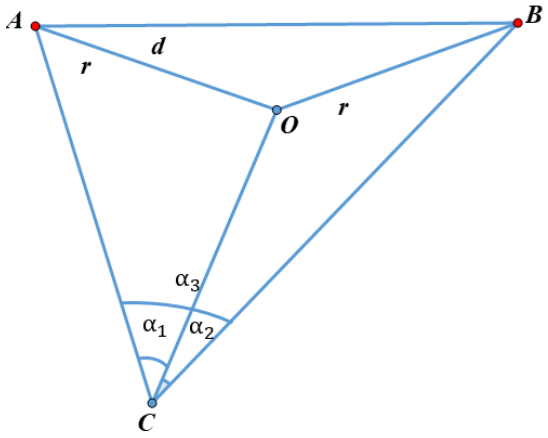


图 1 任意三架飞机示意图

那么，我们能够计算出任意飞机无偏差时，部分定位信息如下表所示：

表 2 部分准确位置角度表

接收信号无人机编号		n_1			n_2	
发射信号的无人机编号	n_0	n_2	n_4	n_0	n_6	n_7
$(\alpha_1,\alpha_2,\alpha_3)$	1.222	0.524	0.698	0.175	0.175	0.349

详细结果见附录 1。

而对于剩余的七架飞机分别能够接收到这三辆发射信号飞机，任一飞机其连线的夹角分别为α_1，α_2，α_3。设编队中发射信号的飞机分别为 A、B 两点，AB 两点距离为 d，中心点到发射信号飞机的距离为 r。我们从九架飞机中任取两架飞机作为我们的发射信号无人机，并将它们的编号分别记为n_a与n_b，它们与编号为n_0的飞机一起构成信号发射飞机系统。我们将被观测飞机的编号数字记为n_c。我们取$min\{n_a、n_b、n_0\}$所对应的飞机所在点与n_c位置的连线记为a，取$max\{n_a、n_b、n_0\}$所对应的飞机所在点与n_c位置的连线记为c，三个点中除最大值与最小值之外，剩下的那个点与n_c位置的连线记为b。我们把a与b的夹角记为α_1，b与c的夹角记为α_2，a与c的夹角记为α_3.

通过分析我们发现，所有的取点情况可以分成两种类别。由于圆周上九个点的位置都是相对的，我们不妨设$n_a = n_1$，我们发现：当$(n_b - 5.5)(n_c - 5.5) \geq 0$时，$\alpha_1 + \alpha_2 = \alpha_3$恒成立。如图 2，当我们取定$n_1$、$n_2$、$n_0$为发射信号飞机，$n_3$为需要测量的位置时，满足$(2 - 5.5)(3 - 5.5) \geq 0$，$\alpha_1 + \alpha_2 = \alpha_3$成立。

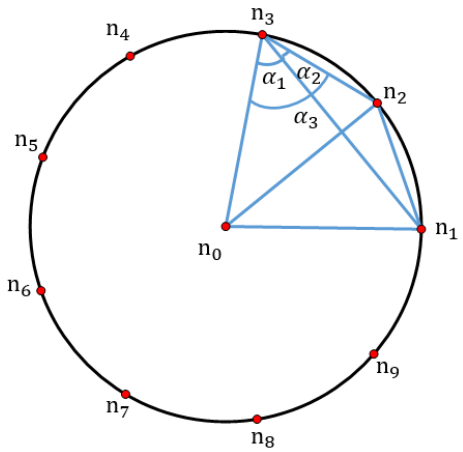


图 2 满足$(n_b - 5.5)(n_c - 5.5) \geq 0$示意图

当$(n_b-5.5)(n_c-5.5)<0$时，$\alpha_1+\alpha_3=\alpha_2$恒成立。如图 3，当我们取定$n_1$、$n_4$、$n_0$为发射信号飞机，$n_7$为需要测量的位置时，满足$(4-5.5)(7-5.5)<0$，$\alpha_1+\alpha_2=\alpha_3$成立。

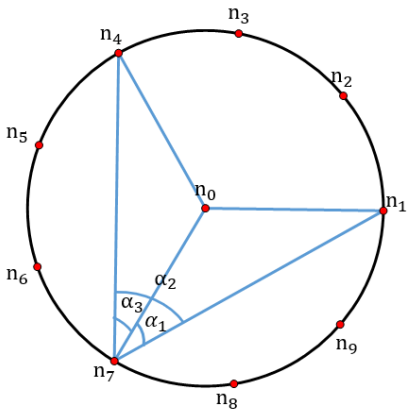


图 3 满足$(n_b-5.5)(n_c-5.5)<0$示意图

设 AC=x，C0=y，CB=z，可以列出方程组

$$\begin{cases} cos\,\alpha_1=\dfrac{x^2+y^2-r^2}{2xy} \\ cos\,\alpha_2=\dfrac{z^2+y^2-r^2}{2zy} \\ cos\,\alpha_3=\dfrac{x^2+z^2-d^2}{2xz} \end{cases} \tag{1}$$

通过方程组，可以求解出这辆飞机到三辆发射信号飞机的距离。示意图如下所示：

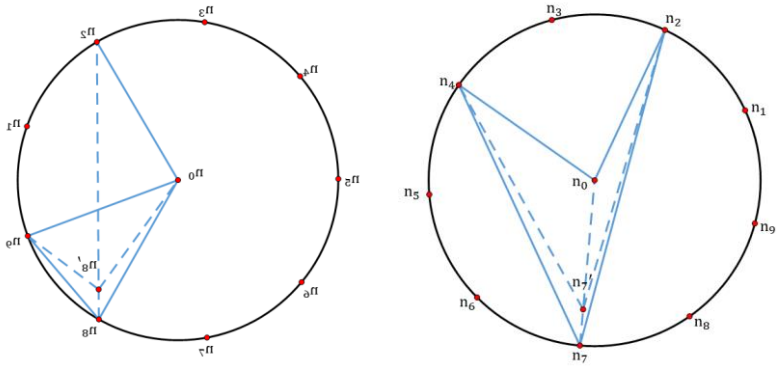


图 4 任意两架发射信号飞机与接收信号飞机示意图

然后利用三角定位法。平面上有三个不共线的发射信号 A,B,O，和一个未知终端 C，并已测出三个基站到终端 C 的距离分别为 x，y，z，则以三个发射信号飞机坐标为圆心，三个点到未知终端距离为半径可以画出三个相交的圆，如图下图所示，需要定位飞机坐标即为三圆相交点Z。

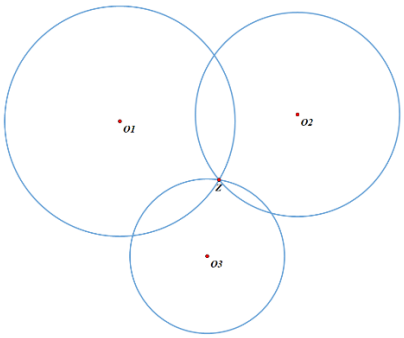


图 5 三个发射信号飞机做圆交点

交点找到后，通过将x、y、z转化为极坐标系，即无人机的准确位置可以用极坐标表示出来。如图 4 任意两架发射信号飞机与接收信号飞机示意图所示：记$\angle AOC = a, \angle COB = b$。

可以得到：

$$\begin{cases} a = arccos\left(\frac{r^2+y^2-x^2}{2ry}\right) \\ b = arccos\left(\frac{r^2+y^2-z^2}{2ry}\right) \end{cases} \tag{2}$$

a、b都可能有两个解，但是必有一个唯一的值，使得其满足等量关系式，从而得到C的极坐标。

5.1.2 第（1）问三角定位模型的求解

我们可以利用三角定位模型，利用 python 采取牛顿迭代法求解被动接收无人机信号的位置。求解步骤如下：

表 3 牛顿迭代法求解三角定位模型步骤表

牛顿迭代法求解三角定位模型步骤：	
Step1:	首先任意给定一组α_1、α_2的角度、发出信号的两架无人机的编号，以及被动接收信号无人机的编号。并确定好中心点发射信号的无人机到其余两架发射信号无人机的距离为r,位于圆周上两架发射信号的无人机间的距离为d。
Step2:	先不考虑接收信号的无人机位置有偏差，在此前提下，利用上述建立的模型，求解出给定被动接收信号无人机飞机与三架发射信号无人机的距离为x_0,y_0,z_0。
Step3:	将上一步得到的x_0,y_0,z_0作为初始条件，代入程序中，不断迭代求解，得到被动接收信号无人机在位置略有偏差时距离发射信号无人机的距离分别为x_1,y_1,z_1。
Step4:	以 FY00 与 FY01 的连线为直径，通过x_1,y_1,z_1的距离求解近似得到的极角，将被动接收信号的无人机用极坐标系表示出来。

通过上述步骤，我们带入了几组数值得到的结果如下表所示：

表 4 第（1）小问部分答案

发射信号无人机		接收信号无人机	极径	极角
FY04	FY05	FY02	1.13	47.17
FY04	FY03	FY05	0.98	162.55
FY07	FY03	FY01	1.09	361.90
FY08	FY07	FY02	0.91	43.47
FY09	FY06	FY03	0.93	74.52

表 5 第（1）小问部分答案

发射信号无人机		接收信号无人机	α1	α2	α3
FY04	FY05	FY02	0.873	0.524	0.249
FY04	FY03	FY05	1.222	0.873	0.409
FY07	FY03	FY01	0.524	0.873	1.376
FY08	FY07	FY02	0.504	0.165	0.369
FY09	FY06	FY03	0.524	0.424	1.047

在发射信号的无人机为n_4与n_5时，我们得到一个n_2附近的一个有略微偏差的点 S，我们取第一组数据，画出与之成比例的图像如下图。

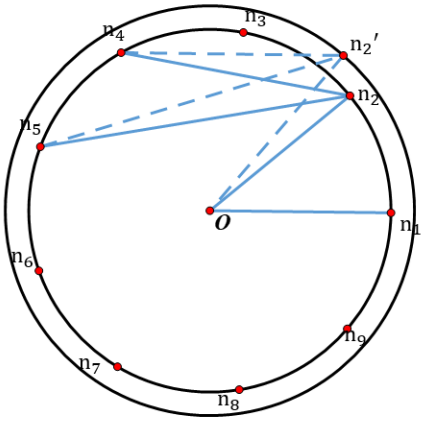


图 6 第一问部分接收信息点图

可行性分析：我们通过题目给出的条件，列出了方程组：

$$\begin{cases} cos\,\alpha_1 = \dfrac{x^2+y^2-r^2}{2xy} \\ cos\,\alpha_2 = \dfrac{z^2+y^2-r^2}{2zy} \\ cos\,\alpha_3 = \dfrac{x^2+z^2-d^2}{2xz} \end{cases} \qquad (3)$$

首先必须肯定，我们的方程组所得解是基于大量事实以及数学知识得到的。而根据实践，我们发现当α_1与α_2以及α_3与理想值差距较大时，我们所得的结果意义不大，而事实表明通过此方程组解出来的数据基本没有异常的波动。

其次，我们通过对数据进行处理，找到了在理想状态下，即所有的无机均在圆周上，然后任意选择圆周上两辆无人机，在不同编号飞机处的信息角度与极径分别为θ_i与ρ_i。同时，我们考虑被测量的飞机在位置上存在偏差的情况，通过相同的方法，求得其角度与极径分别为θ_i'与ρ_i'。

在得到大量数据之后，我们让观测飞机编号相同的理想角度值θ_i与存在偏差的角度值θ_i'以及理想极径ρ_i与存在偏差极径θ_i'做差。我们将它们的差值分别取绝对值，发现误差范围均在0.05%以内。

对于n_7无人机在理想情况下，相对于n_2与n_3的初始角度情况，同时减小了$\alpha_1,\alpha_2,\alpha_3$,并且使它们的减小量也满足$\Delta\alpha_1+\Delta\alpha_2=\Delta\alpha_3$，此时测量点的极径增加，而极角不变，将结果运用几何画板等比例作图后，发现角度基本没有偏差，极径确实增长，符合预期。

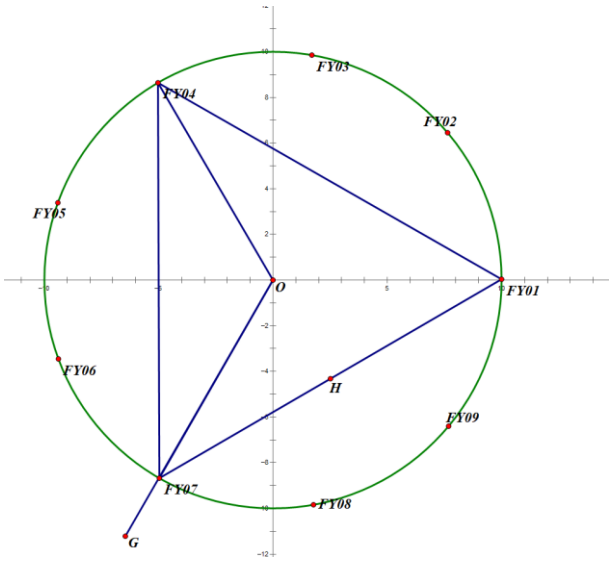


图 7 任给一组数据结果图

5.2 问题一第（2）模型的建立与证明

5.2.1 第（2）问模型的证明

问题二，题目条件告诉我们，两架发射信号的无人机n_0与n_1的位置，而另外编队中发生信号的无人机编号与数量均未知。事实上，我们仍有问题一中的假设，发射信号飞机均分布在一个完整的圆周上，发射信号位置无偏差，呈现出一种比较理想的状态。从已知的两台发射信号的无人机n_0与n_1出发，根据题意，我们可以通过这两台飞机发射的信号，任意选取一架待定位的无人机，记无人机的位置为$n_k(k \neq 0,1)$。在不考虑被接收信号的飞机位置偏移时，如下图所示：

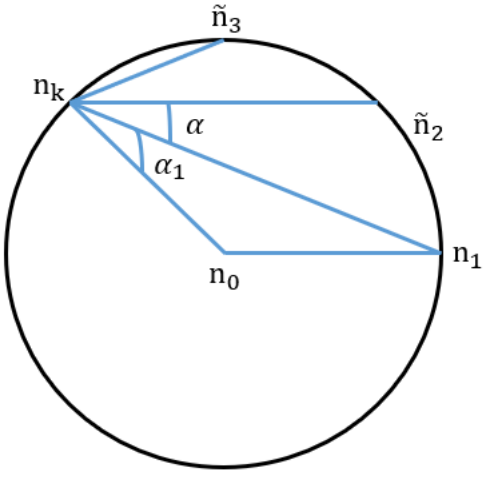


图 8 当不考虑被动接收信号的飞机位置偏移时示意图

我们能知道n_0、n_1、α_1三个数据。如果增加$\widetilde{n_2}$未知编号的无人机发射信号，则可测得$\alpha = \angle\left(\overrightarrow{n_k\, n_0}, \overrightarrow{\widetilde{n_2} n_k}\right)$，因为$\widetilde{n_2}$的编号未知，$\widetilde{n_2}$可能取值有 7 个，即$\widetilde{n_2} = n_i, i = 2,3,\ldots,9,$且$i \neq k$。当每取定一个$\widetilde{n_2}$时，运用第一问的三角定位模型，即可得到 7 个$n_k$的定位点，记这七个定位点集合为$P_k(\alpha)$，必有唯一一个$\gamma \epsilon P_k(\alpha)$，使得$\gamma = n_k$。这时候在增加$\widetilde{n_3}$未知编号的无人机发射信号，可测的$\alpha' = \angle\left(\overrightarrow{n_k\, n_0}, \overrightarrow{\widetilde{n_3} n_k}\right)$。$\widetilde{n_3}$可能取值也有 7 个，即$\widetilde{n_3} = n_i, i = 2,3,\ldots,9,$且$i \neq k$。当每取定一个$\widetilde{n_2}$，同样可得到 7 个$n_k$的定位点，记这七个定位点集合为$P_k(\alpha')$,必有一个$\gamma \epsilon P_k(\alpha)$，使得$\gamma = n_k$。则$P_k(\alpha)$和$P_k(\alpha')$集合中都必存在一个唯一的点，使得

$P_k(\alpha) \cap P_k(\alpha') = n_k$。

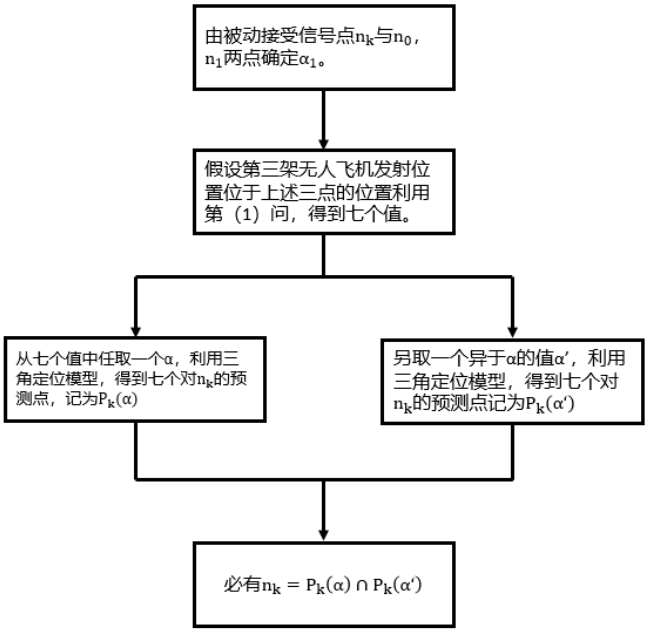


图 9 第（2）小问思路图

5.2.2 第（2）问模型的事例示范

虽然被动接收信号的无人机可能略有偏差，但是点P必定在某一个n_i点附近。不妨设P为n_k的偏差点，$k = 2,3,\ldots,9$。如下图所示，连接n_0p、n_0n_1、n_1p，当P点位置确定时，α_1的角度也确定了。

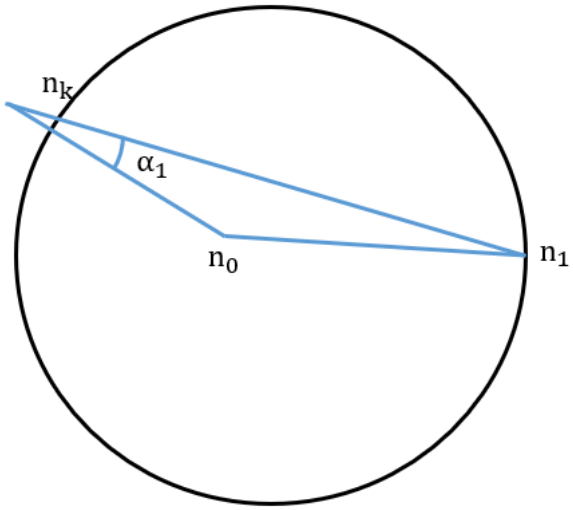


图 10 当被动接收信号无人机位置有偏移时

接着，我们开始增加编队中发射信号无人机的数量。在此情况下，可以发射信号的无人机有7架，令这7架无人机分别发射信号，如果$n_i(i \neq k)$无人机发射信息时，连接发射点与点P，然后我们能够接收到另外一个方向信息γ_i。记集合J_A为：

$$J_A = \{\gamma_2, \gamma_3, \ldots, \gamma_9\}$$

其中，γ_i表示第n_i架无人机发射信号时，其中的一个方向信息，$i = 2,3,\ldots,9$，且$i \neq k$。

然后，我们任取一个$\alpha, \alpha \epsilon J_A$。这个时候，我们就已经得知了$\alpha_1, \alpha$两个方向信息以及$n_0$、$n_1$两个发射信息无人机位置。此时我们增加一个发射信息的无人机，就能根据第一问的三角定位模型求解出P点的近似位置。分别让$n_i, i = 2,3,\dots,9$ 且 i ≠ k，利用模型可以分别得到7个P点的近似位置，记这7个点的集合为Q_P，即：

$$Q_P = \{P_2, P_3, \dots, P_9\}$$

其中，P_i表示第三个发射信号无人机为n_i时，利用三角模型求得的点P的近似位置，$i = 2,3,\dots,9$,且$i \neq k$。

为了找到哪一个无人机作为第三个发射信号点，我们需要再取$\beta, \beta \epsilon J_A/\alpha$。再和上述方法一样，通过第一问三角定位法，分别求解出 7 个 P 点的近似位置，记这 7 个点的集合为$L_{p'}$，即：

$$L_{P'} = \{P_2', P_3', \dots, P_9'\}$$

其中，P_i'表示第三个发射信号无人机为n_i时，利用三角模型求得的点P的近似位置，$i = 2,3,\dots,9$,且$i \neq k$。

这个时候，我们需要去对比两次所求点 P 的近似位置，如果Q_P和$L_{P'}$两个集合中存在两个元素，使得两者距离足够小，那么这两个元素分别对应的无人机位置点即为需增加的无人机信号发射点。即：

$$min\{d(P_i - P_i')\} < \varepsilon \qquad (2)$$

其中，$P_i \in Q_P, P_i' \in L_{P'}$，$d(P_i - P_i')$代表$P_i'$、$P_i$两点间距离，$\varepsilon$为一个极小的数值，由第一问结果分析可知，偏差范围在0.005%之内。

若Q_P、$L_{P'}$中不存在元素满足上述（2）式，那就再任取一个$\rho, \rho \epsilon J_A$，按照上述方法得到一个集合$T_{P''}$，记$T_{P''}$为：

$$T_{P''} = \{P_2'', P_3'', \dots, P_9''\}$$

其中，P_i''表示第三个发射信号无人机为n_i时，利用三角模型求得的点P的近似位置，$i = 2,3,\dots,9$,且$i \neq k$。

取Q_P、$L_{P'}$、$T_{P''}$中三个集合中的两个元素，使得这两个元素的距离最短。然后再看这两点的距离是否小于ε。即：

$$min\{d(P_i - P_i'), d(P_i - P_i''), d(P_i'' - P_i')\} < \varepsilon$$

其中$P_i \in Q_P, P_i' \in L_{P'}$，$P_i'' \in T_{P''}$，$d(P_i - P_i')$、$d(P_i - P_i'')$、$d(P_i'' - P_i')$分别代表$P_i'P_i$、$P_i''P_i$、$P_i'P_i''$两点间距离。

若还不满足条件，那就按照上述方法不断选取点，直到找到满足两点之间距离小于ε的解。找到的满足条件的两个点，即为要增加的发射信号的无人机。

例如，当被动接收信号的无人机为n_4时，根据上文可以马上读取出$\alpha_1 = 30°$。这个时候可以求得

$$J_A = \{20.25°, 40.04°, 50.01°, 70.12°, 29.44°, 9.79°, 9.81°\}$$

从J_A中任取一个元素，不妨取$\alpha = 20.25°$，即我们可以根据$\alpha_1 = 30°$、$\alpha = 20.25°$以及n_i、n_1、n_0三个发射信号无人机，由第一小问模型计算出七个n_4无人机位置的近似值：

$$P_k(\alpha) = \begin{Bmatrix} (7,1.66,80.31), (8,0.95,114.79), (9,-0.03,147.37), \\ (6,1.45,43.453), (5,2.45,12.31), (3,2.17,12.22), (2,1.21,49.92) \end{Bmatrix}$$

其中，三维向量中的元素分别对应飞机编号、极径，极角。

再取$\beta = 40.04°$，从$J_A \backslash A$中任意取一个元素，我们根据$\alpha_1 = 30°$、$\beta = 40.04°$以及n_i、n_1、n_0三个发射信号无人机，由第一小问模型计算出七个n_4无人机位置的近似值：

$$P_k(\beta)=\left\{\begin{matrix}(2,1.13,65.35),(3,1.29,71.34),(5,1.27,102.04)\\(6,1.00,114.80),(7,0.50,130.72),(8,0.34,145.44),(9,0.12,150.46)\end{matrix}\right\}$$

其中，三维向量中的元素分别对应飞机编号、极径，极角。

表 6 当$\alpha=20.25°$、$\beta=40.04°$时$P_k(\alpha)$和$P_k(\beta)$值

$\alpha=20.25°$			$\beta=40.04°$		
飞机编号	极径	极角	飞机编号	极径	极角
2	1.211	49.932	2	1.132	65.345
3	2.116	12.233	3	1.286	71.343
5	5.2445	12.313	5	1.271	102.035
6	1.445	43.453	6	0.954	114.791
7	1.661	80.31	7	0.494	130.716
8	0.954	114.791	8	0.343	145.443
9	−0.027	147.371	9	0.123	150.456

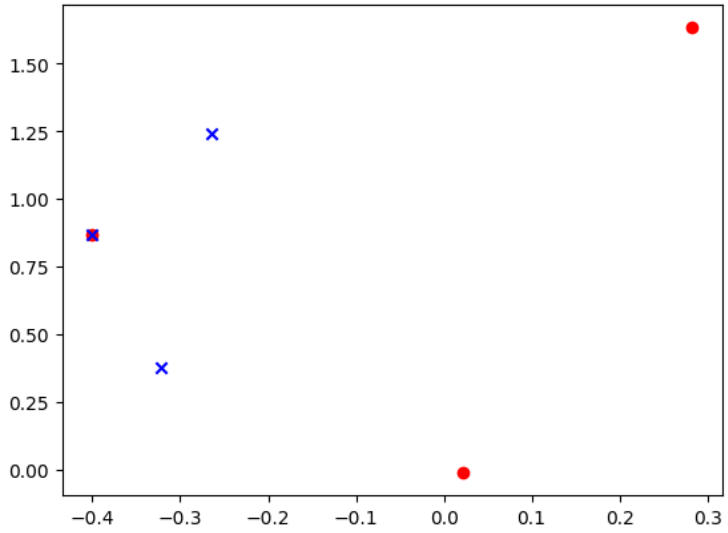


图 11 第二问结果散点图

红色的圆点是选取一个未知编号的发射信号的无人机和 FY00,FY01 后，穷举无人机的编号，计算得出的接收信号的无人机的可能的位置。蓝色的叉点是选取另一个未知编号的发射信号的无人机和 FY00,FY01 后，穷举无人机的编号，计算得出的接收信号的无人机的可能的位置。筛选其中偏离过远的部分结果，画出如上散点图，可以看出在这些数据中只有一个坐标被同时选中，这个坐标便是接受信息的无人机的真实位置，而计算到这个坐标的两个假设发送信息的无人机的编号，便是这两台无人机的真正编号。即$P_k(\alpha)\cap P_k(\beta)$即为我们的答案。即我们需要增加$n_8$与$n_6$两台无人机，才能符合题意。通过大量计算发现，误差在0.02%以内。

5.3 问题一第（3）问模型的建立与求解

5.3.1 基于三角定位法的多目标优化模型

题目要求出来中心点无人机发射信号外，还最多增加3架无人机发射信号，使得其余无人机能够根据接收到的方向信息来调整位置，最终均匀分布在某个圆周上。引入$0-1$决策变量δ_i，描述无人机是否发射信号：

$$\delta_i = \begin{cases} 1,\ n_i\text{无人机发射信号} \\ 0,\ n_i\text{无人机不发射信号} \end{cases} \quad \text{其中，} i = 1,2,3,\dots,9 \qquad (4)$$

由第一问的三点定位法可知，除了圆心一点发射信息外，还需要至少两架无人机发射信息，这样才能使得其余无人机接收到三个方向信息，然后题目限制最多增加3架无人机发射信息。所以，我们得到一个约束条件为：

$$2 \le \sum_{i=1}^{9} \delta_i \le 3 \qquad (5)$$

要使得编队上的九架无人机都均匀的分布在圆周上，根据题目已给的信息，我们可以得到这个圆周的半径为$100m$，那么九个点的理想位置极径都是100，极角如表 2 所示。希望经过调整后，每架无人机都能尽可能的到达自己的理想位置。记：

$$\delta = \{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \delta_7, \delta_8, \delta_9\}$$

其中，δ是一个9×1维向量，δ_1,δ_2,…,δ_9分别代表编号为$n_1, n_2, \dots, n_9$无人机是否发射信号。

知道发射信息的无人机编号后，我们就能根据第一问中的三角定位法，确定点P的实际位置。即：

$$P_i(\delta) = \{r_i(\delta), \theta_i(\delta)\}$$

且$r_i(\delta), \theta_i(\delta)$的关系能满足：

$$\begin{cases} a = arccos\left(\frac{r^2+y^2-x^2}{2ry}\right) \\ b = arccos\left(\frac{r^2+y^2-z^2}{2ry}\right) \end{cases} \qquad (6)$$

于是可建立一个多目标优化模型：

- **极径长度的标准差**

$$min\ Std(r_i(\delta)) = \sqrt{\frac{\sum_{i=1}^{n}\left(r_i(\delta)-\overline{r(\delta)}\right)^2}{n}} \qquad (7)$$

其中，$Std(r_i(\delta))$表示编号n_i无人机实际极径与理想极径的标准差，$r_i(\delta)$表示编号为n_i的无人机的实际极径，$\overline{r(\delta)} = \frac{r_1+r_2+\dots+r_n}{n}$表示每架无人机实际极径的平均值。且$i = 1,2,\dots,9$

- **理想极角差值的标准差**

$$\min\ Std(\theta_i(\delta)) = \sqrt{\frac{\sum_{i=1}^{n}\left(\theta_i(\delta)-\overline{\theta(\delta)}\right)^2}{n}} \qquad (8)$$

其中，$Std(\theta_i(\delta))$表示编号n_i无人机和理想极角差值的标准差，$\theta_i(\delta) = \omega_i - \omega_{i\,accurate}$，$\omega_i$表示编号为$n_i$的无人机的实际极角，$\omega_{i\,accurate}$表示编号$n_i$无人机理想极角，$\overline{\theta(\delta)} = \frac{\theta_1+\theta_2+\dots+\theta_n}{n}$。

综上所述，该多目标优化模型为：

目标函数：

$$\min Std\left(r_i(\delta)\right) = \sqrt{\frac{\sum_{i=1}^{n}\left(r_i(\delta)-\overline{r(\delta)}\right)^2}{n}}$$

$$min\ Std(\theta_i(\delta)) = \sqrt{\frac{\sum_{i=1}^{n}\left(\theta_i(\delta)-\overline{\theta(\delta)}\right)^2}{n}}$$

约束条件：

$$2 \le \sum_{i=1}^{9} \delta_i \le 3$$

为了找到合适的δ_i,模型的优化步骤如下表：

表 7 多目标优化模型步骤

多目标优化模型步骤	
Step1	根据各编号无人机的初始位置，计算目标函数初始值$Std(r_i(\delta))_0, Std(\theta_i(\delta))_0$
Step2	在九架无人机中，选择合适的两架或者三架无人机发射信号，发射信号的无人机位置会略有偏差 其余位置的无人机通过接收方向信息，对自己的位置进行调整。
Step3	每架无人机都根据接收到的方向信息调整到$\frac{\lambda_1\times\text{理想角度}+\lambda_2\times\text{实际角度}}{2}$（$\lambda_1+\lambda_2=1$）
Step4	根据发射信号的三架或者四架飞机，利用第一小问中的三角定位模型，能够求解出每架无人机调整后的具体位置
Step5	计算目标函数值$Std(r_i(\delta))_1, Std(\theta_i(\delta))_1$，如果$Std(r_i(\delta))_0, Std(\theta_i(\delta))_0 > Std(r_i(\delta))_1, Std(\theta_i(\delta))_1$，则更新目标函数值为$Std(r_i(\delta))_0, Std(\theta_i(\delta))_0$

分析得出，发射信号的飞机一共有$C_9^3+C_9^2$种组合，对目标函数不断进行更新，经过$C_9^3+C_9^2$次后，我们最终得到目标函数值为$Std(r_i(\delta))_n, Std(\theta_i(\delta))_n$。如果，

$$\begin{cases} Std(r_i(\delta))_n < \varepsilon_1 \\ Std(\theta_i(\delta))_n < \varepsilon_2 \end{cases} \quad (9)$$

成立。其中，ε_1、ε_2足够小。

说明实际位置与理想位置的偏差极小，达到了我们飞机尽可能的调整到理想位置的目标。同时，我们得到当$P_i(\delta)$的具体数值，知道由哪几架飞机发射信号可以达到目标。

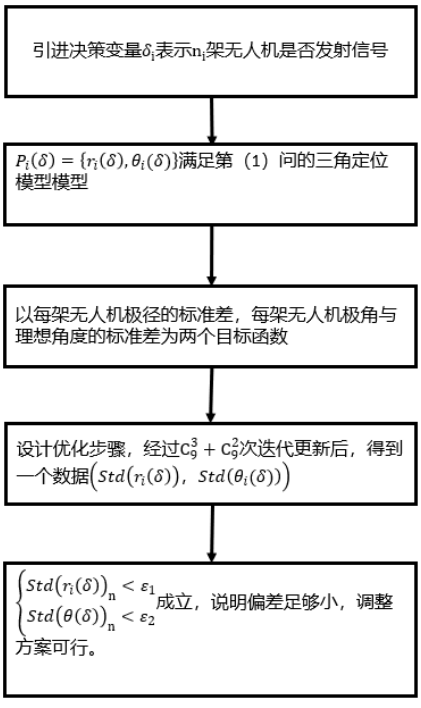


图 12 第（3）小问建立模型思路图

5.3.2 基于三角定位法的多目标优化模型的求解

利用 Python 按照模型，求得每架飞机的具体调整方案。

九架无人机调整的具体方案如下表，最终九架飞机均匀分布在一个极径为104.887的圆周上。

表 8 九架无人机调整方案部分表

	初始	第 1 次迭代	第 2 次迭代
n_0无人机的位置	(0，0)	(0，0)	(0，0)
n_1无人机的位置	(102.53，0.27)	(102.53，0.27)	(102.53，0.27)
n_2无人机的位置	(98.00，40.10)	(103.21，39.91)	(103.21，39.91)
n_3无人机的位置	(112.00，80.21)	(112.00，80.21)	(107.18，80)
n_4无人机的位置	(105.00，119.75)	(105.00，119.75)	(105.00，119.75)
n_5无人机的位置	(98.00，159.86)	(98.00，159.86)	(98.00，159.86)
n_6无人机的位置	(112.00，199.96)	(112.00，199.96)	(112.00，199.96)
n_7无人机的位置	(105.00，240.07)	(105.00，240.07)	(105.00，240.07)
n_8无人机的位置	(98.00，280.17)	(98.00，280.17)	(98.00，280.18)
n_9无人机的位置	(112.00，320.28)	(112.00，320.29)	(112.00，320.30)

表 9 九架无人机调整方案（续表）

	…	第 498 次迭代	第 499 次迭代
n_0无人机的位置	…	(0，0)	(0，0)
n_1无人机的位置	…	(104.89，0.09)	(104.89，0.09)
n_2无人机的位置	…	(104.89，40.09)	(104.89，40.09)
n_3无人机的位置	…	(104.89，80.09)	(104.89，80.09)
n_4无人机的位置	…	(104.89，120.09)	(104.89，120.09)
n_5无人机的位置	…	(104.89，160.09)	(104.89，160.09)
n_6无人机的位置	…	(104.89，200.09)	(104.89，200.09)
n_7无人机的位置	…	(104.89，240.09)	(104.89，240.09)
n_8无人机的位置	…	(104.89，280.09)	(104.89，280.09)
n_9无人机的位置	…	(104.89，320.09)	(104.89，320.09)

详细结果见附件 2。

可行性分析：我们分别利用$Python$做出第 1 次迭代、第 2 次迭代、第 100 次迭代图，如下：

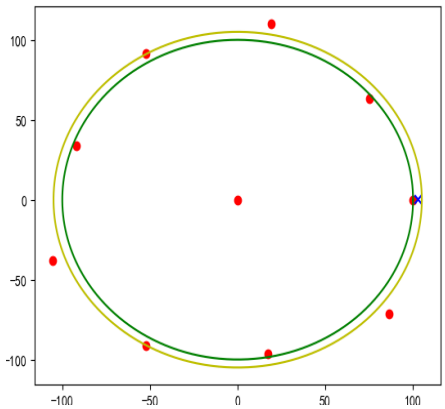


图 13 第一次迭代图

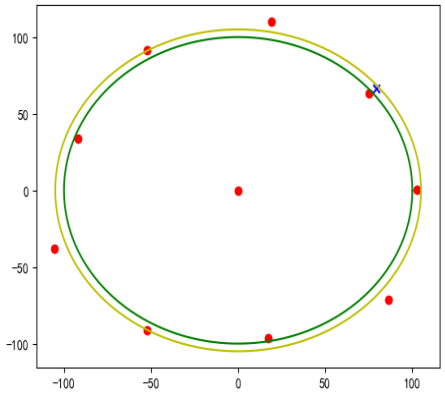


图 14 第二次迭代图

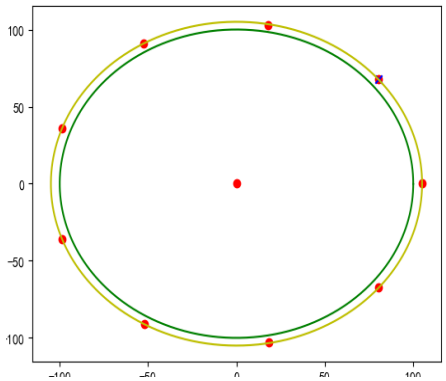


图 15 第 100 次迭代图

上面三个图中，绿色的线代表的是九架无人机组成的理想状态下的圆周，黄色的线代表的是九架无人机经过调整后，组成的一个圆周。红色点表示最开始时每架飞机的偏移位置，蓝色的点表示调整后无人机的位置，可以发现蓝点越来越靠近红点，表明变化越来越小，直至重合。

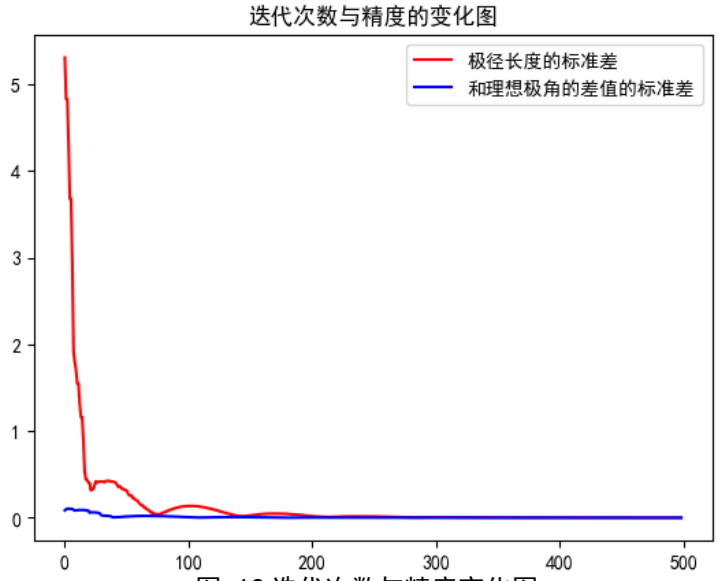


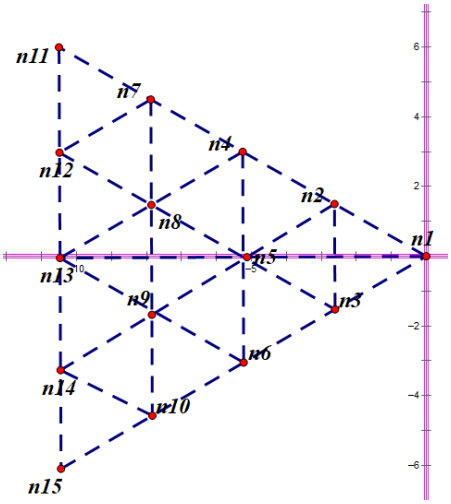
图 16 迭代次数与精度变化图

由上图可以看出，进过调整后，9 架无人机能够均匀分布在一个圆周上。

5. 3 问题二定位模型的建立

5. 3. 1 模型前的准备

当无人机的编队队形为锥形编队队形时，记编号 FY01,FY02,…,FY15 的无人机分别为$n_1, n_2, \ldots, n_{15}$。不妨设n_1无人机的位置是无偏差的，设n_1无人机所在位置为点O，n_{13}无人机位置无偏差的点为Q。以O为原点，OQ连线为x轴，垂直OQ的直线为y轴，建立直角坐标系。如下图所示：



建立直角坐标系后，可以计算出任意三架无人机发射信息时，其余12架无人机在没有偏差时候的精确坐标，如下表所示：

表 10 无人机在没有偏差时的理想坐标

无人机编号	坐标	无人机编号	坐标
n_1	$(0,0)$	n_9	$(-75\sqrt{3},-25)$
n_2	$(-25\sqrt{3},25)$	n_{10}	$(-75\sqrt{3},-75)$
n_3	$(-25\sqrt{3},-25)$	n_{11}	$(-100\sqrt{3},100)$
n_4	$(-50\sqrt{3},50)$	n_{12}	$(-100\sqrt{3},50)$
n_5	$(-50\sqrt{3},0)$	n_{13}	$(-100\sqrt{3},0)$
n_6	$(-50\sqrt{3},-50)$	n_{14}	$(-100\sqrt{3},-50)$
n_7	$(-75\sqrt{3},75)$	n_{15}	$(-100\sqrt{3},-100)$
n_8	$(-75\sqrt{3},25)$		

5.3.2 调整方案的确定

除了n_1无人机外，其余无人机尽可能使得其调整到对应的精确位置上。基于上面几问，规定最多4架无人机遂行发射信号。希望经过调整后，每架无人机都能尽可能的到达自己的理想位置。建立同上文一样的多目标优化模型，决策变量依旧为δ_i，

$$\delta_i=\begin{cases}1,\ n_i\text{无人机发射信号}\\0,\ n_i\text{无人机不发射信号}\end{cases}\qquad \text{其中，}\ i=1,2,3,\dots,15 \qquad (10)$$

但此时的约束条件为：

$$3\le\sum_{i=1}^{15}\delta_i\le 4 \qquad (11)$$

δ也发生改变：

$$\delta=\{\delta_1,\delta_2,\delta_3,\delta_4,\delta_5,\delta_6,\delta_7,\delta_8,\delta_9,\delta_{10},\delta_{11},\delta_{12},\delta_{13},\delta_{14},,\delta_{15}\}$$

其中，δ是一个15×1维向量，$\delta_1,\delta_2,\dots,\delta_{15}$分别代表编号为$n_1,n_2,\dots,n_{15}$无人机是否发射信号。

同样的，知道发射信息的无人机编号后，我们就能根据第一问中的三角定位法，确定点P的实际位置。此时的目标函数依旧为：

$$\min Std\big(r_i(\delta)\big)=\sqrt{\frac{\sum_{i=1}^{n}\left(r_i(\delta)-\overline{r(\delta)}\right)^2}{n}} \qquad (12)$$

$$min\ Std(\theta_i(\delta)) = \sqrt{\frac{\sum_{i=1}^{n}\left(\theta_i(\delta)-\overline{\theta(\delta)}\right)^2}{n}} \qquad (13)$$

其中。$i = 1,2,\ldots,15$。模型的优化步骤还是按照表 7。经过$C_{13}^{2} + C_{13}^{3}$次更新后，使得其余位置无人机最大可能的接近理想状态下的位置。

具体调整方案如下：

首先如图 17 所示，根据上述多目标优化模型，得到n_2, n_3的具体调整方案。

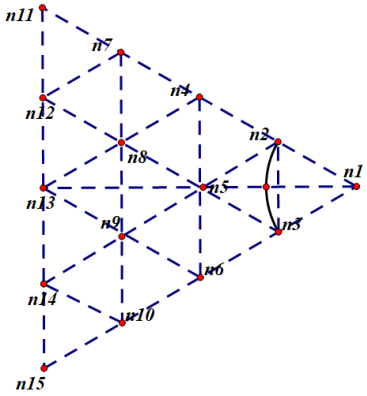


图 17n_2, n_3调整方案

在n_2, n_3优化得到相应位置后，我们可以将n_2, n_3以及n_1作为发射信号的无人机点。如图 18 所示，此时以n_2为圆心，利用上述多目标优化模型，重新得到一组$n_{1,}, n_2, n_4, n_5$无人机调整方案。

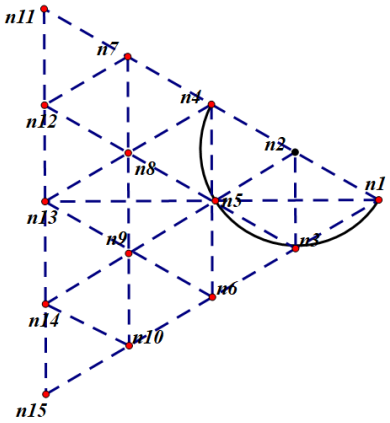


图 18n_4, n_5调整示意图

在$n_{1,}, n_2, n_4, n_5$调整完毕后，以n_5为圆心，利用上述多目标优化模型，调整n_6, n_8, n_9的位置。

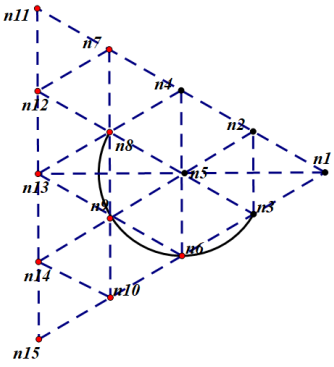


图 19n_6, n_8, n_9调整方案

在n_6, n_8, n_9调整完毕后，以n_8为圆心，利用上述多目标优化模型，调整

n_7, n_{12}, n_{13}的位置。

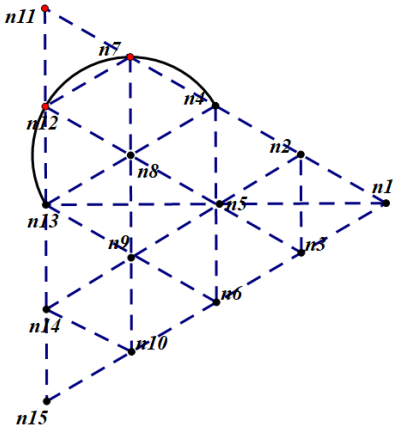


图 20n_7, n_{12}, n_{13}调整方案

在调整完n_7, n_{12}, n_{13}无人机位置后，如下图所示，以n_{12}为圆心，利用上述多目标优化模型，调整n_{11}的位置。

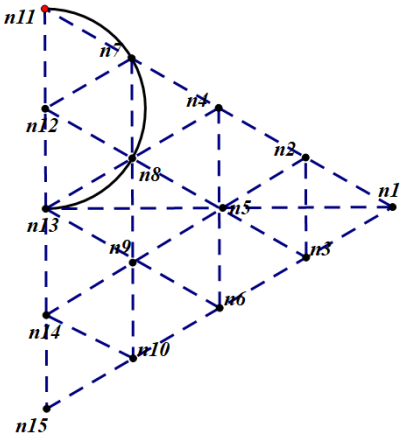


图 21n_{11}调整方案

在调整完n_{11}无人机位置后，如下图所示，以n_{14}为圆心，利用上述多目标优化模型，调整n_{15}的位置。

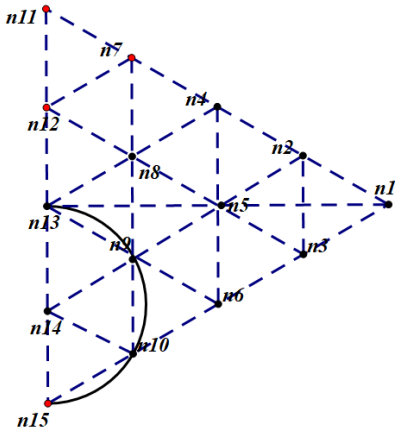


图 22n_{15}调整方案

经过上述六个步骤，通过逐步调整每架无人机的位置，实现15架无人机成一个正三角形的形状。

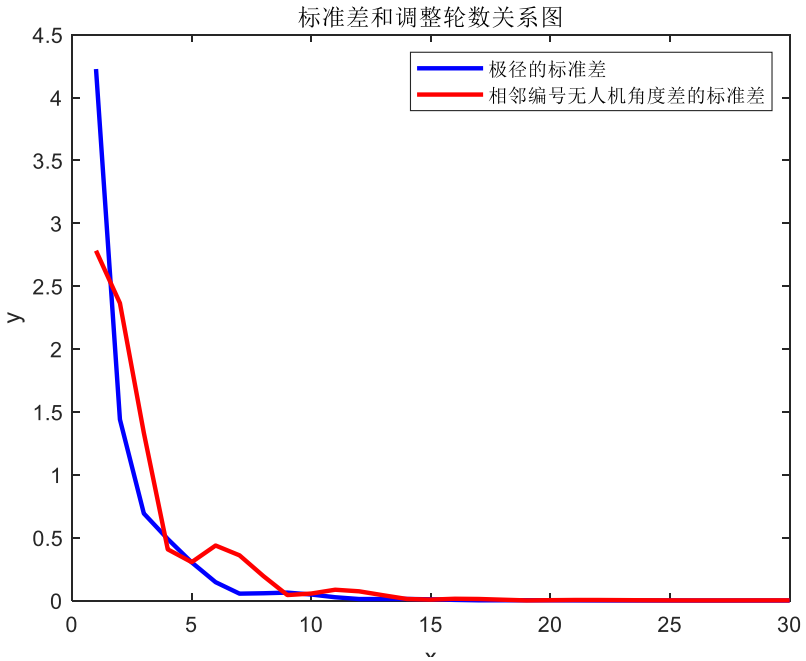


图 23 标准差和调整轮数分析图

我们根据上述模型，进行一个仿真模拟。让无人机的位置在理想状态下随机发生偏移。通过上述模型，将结果不断进行迭代，每次迭代后，都计算出极径标准差和相邻编号无人机的标准差，将结果绘制成图像，发现数据在第 15 次后波动不明显。并且两者的标准差都极小，趋近于0。

六、模型的评价

6.1 模型的优点

第一问第（3）的发送信息的无人机，是选取离当前接收无人机的距离相距最远的两台无人机，选取的标准简单，利于实现。

模型对于初始点的位置具备普适性，对于不同的初始条件，模型可以在较少的迭代次数下完成优化，且极径长度的标准差与迭代次数呈对数分布，表明其优化效率高，且优化后极径长度的标准差与当前极角和理想极角的差值的标准差接近于零，表明优化后无人机均匀分布在一个圆周上。

6.2 模型的缺点

为了解决问题一的模型，需要计算一个三元二次方程组，由于无法解得解析解，便只能使用牛顿迭代法根据初始解，求解数值解，此时结果是局部最优解，而非全局最优解。但数值解的结果与初始解的选取有关系，本文采取使用理想情况下的位置信息计算出的值作为初始解，此时如果实际上的坐标偏离理想过远有可能会导致计算出错误的位置。

对于第一问（3）小问的模型，迭代时计算接收无人机的位置时，调用了问题一的模型，此时假设了两台发送信号的无人机处于理想位置，根据计算的坐标和理想的坐标的偏差值确定优化后的位置，可以看出发送信号的无人机与理想位置的偏差会影响优化后的位置的偏离程度，因此为了排除偶然性的影响，必须大量迭代，最终才能使得结果处于一个较为标准的圆周上。

七、模型的推广

针对无人机的无源定位问题，我们建立了三点定位模型以及优化模型，较好地解决了平面中无源定位问题。通过查阅资料，我们发现对于无人机的定位，无源定位技术不向被定位对象发射电磁信号，这样的特点使得被定位对象不易被感知，减小了被干扰的可能性。因此，在追踪系统的构建过程中，无源定位发挥了决定性的作用。无源位作为迅速发展的电子对抗技术，在电子对抗领域与电子侦察系统里，不可或缺。

在导航系统上，它也发挥了决定性的作用，我们的模型通过四架无人机便一定可以确定任意一辆无人机的位置，这也就意味着，我们不用花费太多成本，就能快速得知任意位置信息。

八、参考文献

[1]郭鹏军,张睿,高关根,许斌.基于相对速度和位置辅助的无人机编队协同导航[J/OL].上海交通大学学报:1-9[2022-09-18].DOI:10.16183/j.cnki.jsjtu.2022.232.

[2]单目标优化，多目标优化，数值优化，组合优化_橘子甜不甜的博客-CSDN博客_单目标优化

[3] 胡来招.《无源定位》.[M] 1-2.国防工业出版社. 2005 年:1-235

[4] https://blog.csdn.net/Reborn_Lee/article/details/82468058

附 录

程序名称	Q1.py	说明	第一问：计算接受信号无人机的位置

```
    import math

import sympy as sp


def func(name_send_1, name_send_2, name_get_1, angle_send1_get1_FY00,
angle_send2_get1_FY00, angle_send1_get1_send2,
         ini, d, r):
    x = sp.Symbol('x')  # 设置符号x
    y = sp.Symbol('y')  # 设置符号y
    z = sp.Symbol('z')  # 设置符号z
    solved_value = sp.nsolve([
        (x ** 2 + z ** 2 - d ** 2) / (2 * x * z) -
sp.cos(angle_send1_get1_send2),
        (x ** 2 + y ** 2 - r ** 2) / (2 * x * y) -
```

```
sp.cos(angle_send1_get1_FY00),
        (y ** 2 + z ** 2 - r ** 2) / (2 * y * z) -
sp.cos(angle_send2_get1_FY00)
    ], [x, y, z], ini)  # 求解
    x = solved_value[0]  # 第一个发信号无人机和接受信号无人机的距离
    y = solved_value[1]  # FY00 和接受信号无人机的距离
    z = solved_value[2]  # 第二个发信号无人机和接受信号无人机的距离

    change = math.acos((y ** 2 + r ** 2 - x ** 2) / (2 * y * r))  # 计算第一个发信号无人机和 FY00 和接受信号无人机的夹角角度
    angle_send1 = (name_send_1 - 1) * math.pi * 4 / 18  # 计算第一个发信号无人机的极角
    angle_send1_change = [(angle_send1 + change + 2 * math.pi) % (2 * math.pi),
                          (angle_send1 - change + 2 * math.pi) % (2 * math.pi)]  # 计算相对于第一个发信号无人机的接受信号无人机的可能位置

    change = math.acos((y ** 2 + r ** 2 - z ** 2) / (2 * y * r))  # 计算第二个发信号无人机和 FY00 和接受信号无人机的夹角角度
    angle_send2 = (name_send_2 - 1) * math.pi * 4 / 18  # 计算第二个发信号无人机的极角

    angle_send2_change = [(angle_send2 + change + 2 * math.pi) % (2 * math.pi),
                          (angle_send2 - change + 2 * math.pi) % (2 * math.pi)]  # 计算相对于第二个发信号无人机的接受信号无人机的可能位置

    ans = 0
    for i in angle_send1_change:  # 从可能无人机角度中寻找相同的角度，这个角度就是接受信号的无人机的真实极角
        for j in angle_send2_change:
            if (math.fabs(i - j) < 0.6):
                ans = (i + j) / 2
            elif (math.fabs(i - j) > 6):
                ans = (i + j + 2 * math.pi) / 2

    return (y, ans / math.pi * 180)  # 返回接受信号的无人机的极坐标


def fift(name_send_1, name_send_2, name_get_1, angle_send1_get1_FY00, angle_send2_get1_FY00, angle_send1_get1_send2):
    r = 1  # 无人机所在的圆的半径
    name_send_1 = int(str(name_send_1).split('FY')[1])  # 处理编号
    name_send_2 = int(str(name_send_2).split('FY')[1])  # 处理编号
```

```
    name_get_1 = int(str(name_get_1).split('FY')[1])  # 处理编号
    ct = min([math.fabs(name_send_1 - name_send_2),
math.fabs(name_send_2 - name_send_1),
              math.fabs(name_send_2 - 9 - name_send_1),
math.fabs(name_send_1 + 9 - name_send_2)])
    ct = ct * math.pi * 4 / 18  # 第一个发信号无人机和FY00和第二个发信号无人机的角的角度
    d = math.sqrt(2 * r * r - 2 * r * r * math.cos(ct))  # 第一个发信号无人机和第二个发信号无人机的距离
    ini = []  # 初始解
    ct = min([math.fabs(name_send_1 - name_get_1), math.fabs(name_get_1
- name_send_1),
              math.fabs(name_send_1 - 9 - name_get_1),
math.fabs(name_get_1 + 9 - name_send_1)])
    ct = ct * math.pi * 4 / 18  # 理想情况下第一个发信号无人机和接受信号的无人机和FY00的角的角度
    ini.append(math.sqrt(2 * r * r - 2 * r * r * math.cos(ct)))  # 理想情况下第一个发信号无人机和接受信号无人机的距离
    ini.append(r)  # 理想情况下FY00和接受信号无人机的距离
    ct = min([math.fabs(name_send_2 - name_get_1), math.fabs(name_get_1
- name_send_2),
              math.fabs(name_send_2 - 9 - name_get_1),
math.fabs(name_get_1 + 9 - name_send_1)])
    ct = ct * math.pi * 4 / 18  # 理想情况下第一个发信号无人机和接受信号的无人机和FY00的角的角度
    ini.append(math.sqrt(2 * r * r - 2 * r * r * math.cos(ct)))  # 理想情况下第一个发信号无人机和接受信号无人机的距离
    return func(name_send_1, name_send_2, name_get_1,
angle_send1_get1_FY00, angle_send2_get1_FY00,
                angle_send1_get1_send2, ini, d, r)  # 计算接受信号的无人机极坐标


if __name__ == '__main__':
    name_send1 = 'FY09'  # 第一个发信号无人机的编号
    name_send2 = 'FY06'  # 第二个发信号无人机的编号
    name_get1 = 'FY03'  # 接受信号的无人机的编号
    angle_send1_get1_FY00 = 0.524  # 第一个发信号无人机与接受信号的无人机和FY00形成的角度
    angle_send2_get1_FY00 = 0.424  # 第二个发信号无人机与接受信号的无人机和FY00形成的角度
    angle_send1_get1_send2 = 1.047  # 第一个发信号无人机与接受信号的无人机和第二个发信号无人机形成的角度
    print(fift(name_send1, name_send2, name_get1, angle_send1_get1_FY00,
```

```
angle_send2_get1_FY00,
                angle_send1_get1_send2))  # 输出接受信号的无人机极坐标
```

程序名称	Q2.py	说明	第二问：计算接受信息的无人机的位置

```
    import math

import sympy as sp


def do_table():  # 返回理想情况下的角度表格
    a = []  # 理想情况下的坐标
    for i in range(9):
        a.append([math.cos(math.pi * i * 4 / 18), math.sin(math.pi * i *
4 / 18)])  # 添加理想点坐标
    table = []
    for i in range(9):  # 穷举第一台发送信号的无人机的编号
        send1 = []
        for j in range(9):  # 穷举第二台发送信号的无人机的编号
            send2 = []
            for k in range(9):  # 穷举接受信号的无人机的编号
                if (i == k or k == j or i == j):  # 处理编号重复的情况
                    send2.append([0, 0, 0])
                    continue
                send3 = []
                send3.append(math.acos(math.sqrt(
                    (a[i][0] - a[k][0]) * (a[i][0] - a[k][0]) + (a[i][1]
- a[k][1]) * (a[i][1] - a[k][1])) / 2))
                # 计算第一个发信号无人机与接受信号的无人机和 FY00 形成的
角度
                send3.append(math.acos(math.sqrt((a[j][0] - a[k][0]) *
(a[j][0] - a[k][0]) + (
                        a[j][1] - a[k][1]) * (a[j][1] - a[k][1])) / 2))
                # 计算第二个发信号无人机与接受信号的无人机和 FY00 形成的
角度
                send3.append(math.acos((math.pow(a[i][0] - a[k][0], 2) +
math.pow(a[i][1] - a[k][1], 2) + math.pow(
                    a[j][0] - a[k][0], 2) + math.pow(a[j][1] - a[k][1],
2) - math.pow(a[i][0] - a[j][0], 2) - math.pow(
                    a[i][1] - a[j][1], 2)) / (2 * math.sqrt(
                    math.pow(a[i][0] - a[k][0], 2) + math.pow(a[i][1] -
```

```
a[k][1], 2)) * math.sqrt(
                    math.pow(a[j][0] - a[k][0], 2) + math.pow(a[j][1] -
a[k][1], 2)))))
                # 计算第一个发信号无人机与接受信号的无人机和第二个发信号
无人机形成的角度
                send2.append(send3)
            send1.append(send2)
        table.append(send1)
    return table


def func(name_send_1, name_send_2, name_get_1, angle_send1_get1_FY00,
angle_send2_get1_FY00, angle_send1_get1_send2,
         ini, d, r):
    x = sp.Symbol('x')  # 设置符号x
    y = sp.Symbol('y')  # 设置符号y
    z = sp.Symbol('z')  # 设置符号z
    solved_value = sp.nsolve([
        (x ** 2 + z ** 2 - d ** 2) / (2 * x * z) -
sp.cos(angle_send1_get1_send2),
        (x ** 2 + y ** 2 - r ** 2) / (2 * x * y) -
sp.cos(angle_send1_get1_FY00),
        (y ** 2 + z ** 2 - r ** 2) / (2 * y * z) -
sp.cos(angle_send2_get1_FY00)
    ], [x, y, z], ini)  # 求解
    x = solved_value[0]  # 第一个发信号无人机和接受信号无人机的距离
    y = solved_value[1]  # FY00和接受信号无人机的距离
    z = solved_value[2]  # 第二个发信号无人机和接受信号无人机的距离

    change = math.acos((y ** 2 + r ** 2 - x ** 2) / (2 * y * r))  # 计算
第一个发信号无人机和FY00和接受信号无人机的夹角角度
    angle_send1 = (name_send_1 - 1) * math.pi * 4 / 18  # 计算第一个发信
号无人机的极角
    angle_send1_change = [(angle_send1 + change + 2 * math.pi) % (2 *
math.pi),
                          (angle_send1 - change + 2 * math.pi) % (2 *
math.pi)]  # 计算相对于第一个发信号无人机的接受信号无人机的可能位置

    change = math.acos((y ** 2 + r ** 2 - z ** 2) / (2 * y * r))  # 计算
第二个发信号无人机和FY00和接受信号无人机的夹角角度
    angle_send2 = (name_send_2 - 1) * math.pi * 4 / 18  # 计算第二个发信
号无人机的极角

    angle_send2_change = [(angle_send2 + change + 2 * math.pi) % (2 *
```

```
math.pi),
                          (angle_send2 - change + 2 * math.pi) % (2 *
math.pi)]  # 计算相对于第二个发信号无人机的接受信号无人机的可能位置

    ans = 0
    for i in angle_send1_change:  # 从可能无人机角度中寻找相同的角度，这
个角度就是接受信号的无人机的真实极角
        for j in angle_send2_change:
            if (math.fabs(i - j) < 0.6):
                ans = (i + j) / 2
            elif (math.fabs(i - j) > 6):
                ans = (i + j + 2 * math.pi) / 2

    return (y, ans / math.pi * 180)  # 返回接受信号的无人机的极坐标


def Q1_solve(name_send_1, name_send_2, name_get_1,
angle_send1_get1_FY00, angle_send2_get1_FY00,
             angle_send1_get1_send2):
    r = 1  # 无人机所在的圆的半径
    name_send_1 = int(str(name_send_1).split('FY')[1])  # 处理编号
    name_send_2 = int(str(name_send_2).split('FY')[1])  # 处理编号
    name_get_1 = int(str(name_get_1).split('FY')[1])  # 处理编号
    ct = min([math.fabs(name_send_1 - name_send_2),
math.fabs(name_send_2 - name_send_1),
              math.fabs(name_send_2 + 9 - name_send_1),
math.fabs(name_send_1 - 9 - name_send_2)])
    ct = ct * math.pi * 4 / 18  # 第一个发信号无人机和FY00和第二个发信
号无人机的角的角度
    d = math.sqrt(2 * r * r - 2 * r * r * math.cos(ct))  # 第一个发信号
无人机和第二个发信号无人机的距离
    ini = []  # 初始解
    ct = min([math.fabs(name_send_1 - name_get_1), math.fabs(name_get_1
- name_send_1),
              math.fabs(name_send_1 + 9 - name_get_1),
math.fabs(name_get_1 - 9 - name_send_1)])
    ct = ct * math.pi * 4 / 18  # 理想情况下第一个发信号无人机和接受信号
的无人机和FY00的角的角度
    ini.append(math.sqrt(2 * r * r - 2 * r * r * math.cos(ct)))  # 理想
情况下第一个发信号无人机和接受信号无人机的距离
    ini.append(r)  # 理想情况下FY00和接受信号无人机的距离
    ct = min([math.fabs(name_send_2 - name_get_1), math.fabs(name_get_1
- name_send_2),
              math.fabs(name_send_2 + 9 - name_get_1),
```

```
math.fabs(name_get_1 - 9 - name_send_1)])
    ct = ct * math.pi * 4 / 18  # 理想情况下第一个发信号无人机和接受信号的无人机和FY00的角的角度
    ini.append(math.sqrt(2 * r * r - 2 * r * r * math.cos(ct)))  # 理想情况下第一个发信号无人机和接受信号无人机的距离
    return func(name_send_1, name_send_2, name_get_1,
angle_send1_get1_FY00, angle_send2_get1_FY00,
                angle_send1_get1_send2, ini, d, r)  # 计算接受信号的无人机极坐标


def Q2_solve(angle_01, angle_0n, angle_1n, angle_0m, angle_1m,
name_get_1):
    table = do_table()  # 计算理想情况下各个无人机之间形成的夹角
    array_n = []  # 穷举第一个未知编号的无人机的的编号后得到的接受信号的无人机的可能的位置
    array_m = []  # 穷举第二个未知编号的无人机的的编号后得到的接受信号的无人机的可能的位置
    self = int(str(name_get_1).split('FY')[1]) - 1  # 处理接受信号的无人机的编号
    for i in range(1, 9):  # 穷举可能的无人机的编号
        name_send_n = 'FY0' + str(i + 1)  # 形成编号
        if (name_send_n == name_get_1 or math.fabs(table[0][i][self][0]
- angle_01) > 0.5 or math.fabs(
                table[0][i][self][1] - angle_0n) > 0.5 or
math.fabs(table[0][i][self][2] - angle_1n) > 0.5):
            continue
        # 处理偏差过大的情况，或者编号冲突的情况

        array_n.append(
            (i + 1, Q1_solve('FY01', name_send_n, name_get_1, angle_01,
angle_0n, angle_1n)))  # 记录计算的接受信号的无人机的可能坐标
    for i in range(1, 9):  # 穷举可能的无人机的编号
        name_send_m = 'FY0' + str(i + 1)  # 形成编号
        if (name_send_m == name_get_1 or math.fabs(table[0][i][self][0]
- angle_01) > 0.5 or math.fabs(
                table[0][i][self][1] - angle_0m) > 0.5 or
math.fabs(table[0][i][self][2] - angle_1m) > 0.5):
            continue
        # 处理偏差过大的情况，或者编号冲突的情况
        array_m.append(
            (i + 1, Q1_solve('FY01', name_send_m, name_get_1, angle_01,
angle_0m, angle_1m)))  # 记录计算的接受信号的无人机的可能坐标
```

```
    for i in array_n:  # 遍历可能的坐标点，最接近的两个坐标的就是接受无人机的坐标
        for j in array_m:
            if (math.fabs(i[1][0] - j[1][0]) <= 0.3):
                return_len = (i[1][0] + j[1][0]) / 2

                if (math.fabs(i[1][1] - j[1][1]) <= 10):
                    return_angle = (i[1][1] + j[1][1]) / 2

                    return (i[0], j[0], return_len, return_angle)

                elif (math.fabs(i[1][1] - j[1][1]) >= 350):
                    return_angle = (i[1][1] + j[1][1]) / 2 + 180

                    return (i[0], j[0], return_len, return_angle)

            elif (math.fabs(i[1][0] - j[1][0]) >= 5.8):
                return_len = (i[1][0] + j[1][0]) / 2 + math.pi
                if (math.fabs(i[1][1] - j[1][1]) <= 10):
                    return_angle = (i[1][1] + j[1][1]) / 2

                    return (i[0], j[0], return_len, return_angle)

                elif (math.fabs(i[1][1] - j[1][1]) >= 350):
                    return_angle = (i[1][1] + j[1][1]) / 2 + 180

                    return (i[0], j[0], return_len, return_angle)  # 返回两家未知编号的无人机的编号和接收信号的无人机的位置


if __name__ == "__main__":
    angle_FY01_get1_FY00 = 0.584  # 第一个FY01与接受信号的无人机和FY00形成的角度
    angle_send1_get1_FY00 = 0.132  # 第一个未知位置的发信号无人机与接受信号的无人机和FY00形成的角度
    angle_send1_get1_FY01 = 0.716  # 第一个未知位置的发信号无人机与接受信号的无人机和FY01形成的角度
    angle_send2_get1_FY00 = 0.852  # 第二个未知位置的发信号无人机与接受信号的无人机和FY00形成的角度
    angle_send2_get1_FY01 = 1.436  # 第二个未知位置的发信号无人机与接受信号的无人机和Fy01形成的角度
    name_get = 'FY04'  # 接受信号的无人机编号
    print(Q2_solve(angle_FY01_get1_FY00, angle_send1_get1_FY00, angle_send1_get1_FY01, angle_send2_get1_FY00,
```

```
                    angle_send2_get1_FY01, name_get))  # 计算两家未知编号的无人机的编号和接收信号的无人机的位置
```

程序名称	Q3. py	说明	第三问：对无人机位置优化并展示精度

```
    import cmath
import math

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sympy as sp


def do_table():  # 返回理想情况下的角度表格
    a = []  # 理想情况下的坐标
    for i in range(9):
        a.append([math.cos(math.pi * i * 4 / 18), math.sin(math.pi * i * 4 / 18)])  # 添加理想点坐标
    table = []
    for i in range(9):  # 穷举第一台发送信号的无人机的编号
        send1 = []
        for j in range(9):  # 穷举第二台发送信号的无人机的编号
            send2 = []
            for k in range(9):  # 穷举接受信号的无人机的编号
                if (i == k or k == j or i == j):  # 处理编号重复的情况
                    send2.append([0, 0, 0])
                    continue
                send3 = []
                send3.append(math.acos(math.sqrt(
                    (a[i][0] - a[k][0]) * (a[i][0] - a[k][0]) + (a[i][1] - a[k][1]) * (a[i][1] - a[k][1])) / 2))
                # 计算第一个发信号无人机与接受信号的无人机和 FY00 形成的角度
                send3.append(math.acos(math.sqrt((a[j][0] - a[k][0]) * (a[j][0] - a[k][0]) + (
                        a[j][1] - a[k][1]) * (a[j][1] - a[k][1])) / 2))
                # 计算第二个发信号无人机与接受信号的无人机和 FY00 形成的角度
```

```
                send3.append(math.acos((math.pow(a[i][0] - a[k][0], 2) +
math.pow(a[i][1] - a[k][1], 2) + math.pow(
                    a[j][0] - a[k][0], 2) + math.pow(a[j][1] - a[k][1],
2) - math.pow(a[i][0] - a[j][0], 2) - math.pow(
                    a[i][1] - a[j][1], 2)) / (2 * math.sqrt(
                    math.pow(a[i][0] - a[k][0], 2) + math.pow(a[i][1] -
a[k][1], 2)) * math.sqrt(
                    math.pow(a[j][0] - a[k][0], 2) + math.pow(a[j][1] -
a[k][1], 2)))))
                # 计算第一个发信号无人机与接受信号的无人机和第二个发信号
无人机形成的角度
                send2.append(send3)
            send1.append(send2)
        table.append(send1)
    return table


def answer(name_send_1, name_send_2, name_get_1, angle_1, angle_2,
angle_3, ini, d, r):
    x = sp.Symbol('x')# 设置符号x
    y = sp.Symbol('y')# 设置符号y
    z = sp.Symbol('z')# 设置符号z
    solved_value = sp.nsolve([
        (x ** 2 + z ** 2 - d ** 2) - math.cos(angle_3) * (2 * x * z),
        (x ** 2 + y ** 2 - r ** 2) - math.cos(angle_1) * (2 * x * y),
        (y ** 2 + z ** 2 - r ** 2) - math.cos(angle_2) * (2 * y * z)
    ], [x, y, z], ini, prec=10)# 求解

    x = solved_value[0]# 第一个发信号无人机和接受信号无人机的距离
    y = solved_value[1]# FY00和接受信号无人机的距离
    z = solved_value[2]# 第二个发信号无人机和接受信号无人机的距离

    change = math.acos((y ** 2 + r ** 2 - x ** 2) / (2 * y * r))# 计算第
一个发信号无人机和FY00和接受信号无人机的夹角角度
    a = (name_send_1 - 1) * math.pi * 4 / 18# 计算第一个发信号无人机的极
角
    a_change = [(a + change + 2 * math.pi) % (2 * math.pi), (a - change
+ 2 * math.pi) % (2 * math.pi)]# 计算相对于第一个发信号无人机的接受信号
无人机的可能位置
    change = math.acos((y ** 2 + r ** 2 - z ** 2) / (2 * y * r))# 计算第
二个发信号无人机和FY00和接受信号无人机的夹角角度
    b = (name_send_2 - 1) * math.pi * 4 / 18# 计算第二个发信号无人机的极
角
    b_change = [(b + change + 2 * math.pi) % (2 * math.pi), (b - change
```

```
+ 2 * math.pi) % (2 * math.pi)] # 计算相对于第二个发信号无人机的接受信号无人机的可能位置
    ans = 0
    for i in a_change:# 从可能无人机角度中寻找相同的角度，这个角度就是接受信号的无人机的真实极角
        for j in b_change:
            if (math.fabs(i - j) < 0.6):
                ans = (i + j) / 2
            elif (math.fabs(i - j) > 5.7):
                ans = (i + j + 2 * math.pi) / 2

    return (y * math.cos(ans), y * math.sin(ans))# 返回接受信号的无人机的极坐标


def Q1_solve(name_send_1, name_send_2, name_get_1, angle_1, angle_2, angle_3):
    r = 100  # 无人机所在的圆的半径
    name_send_1 = int(str(name_send_1).split('FY')[1])  # 处理编号
    name_send_2 = int(str(name_send_2).split('FY')[1])  # 处理编号
    name_get_1 = int(str(name_get_1).split('FY')[1])  # 处理编号

    ct = min([math.fabs(name_send_1 - name_send_2), math.fabs(name_send_2 - name_send_1),
              math.fabs(name_send_2 - 9 - name_send_1), math.fabs(name_send_1 + 9 - name_send_2)])

    ct = ct * math.pi * 4 / 18  # 第一个发信号无人机和FY00和第二个发信号无人机的角的角度

    d = math.sqrt(2 * r * r - 2 * r * r * math.cos(ct))  # 第一个发信号无人机和第二个发信号无人机的距离
    ini = []  # 初始解
    ct = min([math.fabs(name_send_1 - name_get_1), math.fabs(name_get_1 - name_send_1),
              math.fabs(name_send_1 - 9 - name_get_1), math.fabs(name_get_1 + 9 - name_send_1)])

    ct = ct * math.pi * 4 / 18  # 理想情况下第一个发信号无人机和接受信号的无人机和FY00的角的角度
    ini.append(math.sqrt(2 * r * r - 2 * r * r * math.cos(ct)))
    ini.append(r)
    ct = min([math.fabs(name_send_2 - name_get_1), math.fabs(name_get_1 - name_send_2),
```

```
                math.fabs(name_send_2 - 9 - name_get_1),
math.fabs(name_get_1 + 9 - name_send_1)])
    # print(ct)
    ct = ct * math.pi * 4 / 18  # 理想情况下第一个发信号无人机和接受信号的无人机和FY00的角的角度

    ini.append(math.sqrt(2 * r * r - 2 * r * r * math.cos(ct)))  # 理想情况下第一个发信号无人机和接受信号无人机的距离

    return answer(name_send_1, name_send_2, name_get_1, angle_1,
angle_2, angle_3, ini, d, r)  # 计算接受信号的无人机极坐标


def Q3_solve(Polar):
    r = 100  # 无人机所在的圆的半径
    table = do_table()  # 计算理想情况下各个无人机之间形成的夹角
    Polar = Polar[1:]
    Ideal_location = [(0, 0)]  # 理想情况下点的直角坐标系坐标
    for i in range(1, 10):
        Ideal_location.append((r * math.cos(math.pi * (i - 1) * 40 /
180),
                               r * math.sin(math.pi * (i - 1) * 40 /
180)))  # 计算理想情况下点的直角坐标系坐标
    Right_Angle = np.zeros((10, 2)).tolist()  # 将实际无人机位置的极坐标系坐标转换为直角坐标坐标
    for i in range(1, 10):
        Right_Angle[i][0] = Polar[i - 1][0] * math.cos(Polar[i - 1][1] /
180 * math.pi)
        Right_Angle[i][1] = Polar[i - 1][0] * math.sin(Polar[i - 1][1] /
180 * math.pi)
        # 将实际无人机位置的极坐标系坐标转换为直角坐标坐标
    get_1 = 0  # 记录接受的无人机编号
    ans = []
    for _ in range(500):  # 循环固定次数

        i = (get_1 + 4) % 9 + 1  # 第一台发送信号的无人机的编号
        j = (get_1 + 5) % 9 + 1  # 第二台发送信号的无人机的编号
        k = get_1 + 1  # 接收信号无人机的编号
        get_1 = (get_1 + 1) % 9  # 更新接受信号的无人机编号

        angle_ik = math.acos((math.pow(Right_Angle[i][0] -
Right_Angle[k][0], 2) + math.pow(
            Right_Angle[i][1] - Right_Angle[k][1], 2) + math.pow(
            Right_Angle[k][0], 2) + math.pow(Right_Angle[k][1], 2) -
```

```
math.pow(Right_Angle[i][0], 2) - math.pow(
            Right_Angle[i][1], 2)) / (2 * math.sqrt(
            math.pow(Right_Angle[i][0] - Right_Angle[k][0], 2) +
math.pow(Right_Angle[i][1] - Right_Angle[k][1],

2)) * math.sqrt(
            math.pow(Right_Angle[k][0], 2) + math.pow(Right_Angle[k][1],
2))))
        angle_ik = (angle_ik + table[i - 1][j - 1][k - 1][0]) / 2  # 第
一台发信号无人机与接受信号无人机与FY00间的角度

        angle_jk = math.acos((math.pow(Right_Angle[j][0] -
Right_Angle[k][0], 2) + math.pow(
            Right_Angle[j][1] - Right_Angle[k][1], 2) + math.pow(
            Right_Angle[k][0], 2) + math.pow(Right_Angle[k][1], 2) -
math.pow(Right_Angle[j][0], 2) - math.pow(
            Right_Angle[j][1], 2)) / (2 * math.sqrt(
            math.pow(Right_Angle[j][0] - Right_Angle[k][0], 2) +
math.pow(Right_Angle[j][1] - Right_Angle[k][1],

2)) * math.sqrt(
            math.pow(Right_Angle[k][0], 2) + math.pow(Right_Angle[k][1],
2))))
        angle_jk = (angle_jk + table[i - 1][j - 1][k - 1][1]) / 2  # 第
二台发信号无人机与接受信号无人机与FY00间的角度

        angle_ij = math.acos((math.pow(Right_Angle[i][0] -
Right_Angle[k][0], 2) + math.pow(
            Right_Angle[i][1] - Right_Angle[k][1], 2) + math.pow(
            Right_Angle[j][0] - Right_Angle[k][0], 2) +
math.pow(Right_Angle[j][1] - Right_Angle[k][1], 2) - math.pow(
            Right_Angle[i][0] - Right_Angle[j][0], 2) - math.pow(
            Right_Angle[i][1] - Right_Angle[j][1], 2)) / (2 * math.sqrt(
            math.pow(Right_Angle[i][0] - Right_Angle[k][0], 2) +
math.pow(
                Right_Angle[i][1] - Right_Angle[k][1], 2)) * math.sqrt(
            math.pow(Right_Angle[j][0] - Right_Angle[k][0], 2) +
math.pow(Right_Angle[j][1] - Right_Angle[k][1], 2))))
        angle_ij = (angle_ij + table[i - 1][j - 1][k - 1][2]) / 2  # 第
一台发信号无人机与接受信号无人机与第二台发信号无人机间的角度

        A = Q1_solve('FY0' + str(i), 'FY0' + str(j), 'FY0' + str(k),
angle_ik, angle_jk, angle_ij)  # 解出接受信号的无人机坐标
        Right_Angle[k][0] += Ideal_location[k][0] - A[0]
```

```
        Right_Angle[k][1] += Ideal_location[k][1] - A[1]
        # 进行坐标变换
        jzb = []
        for i in Right_Angle:
            jzb.append(cmath.polar(complex(i[0], i[1])))
        for i in range(10):
            jzb[i] = [list(jzb[i])[0], list(jzb[i])[1] * 180 / math.pi]
        ans.append(jzb)
        # 记录极坐标

    for i in ans:
        for j in range(6, 10):
            i[j][1] += 360
            # 将负角度变为正角度
    pd.DataFrame(ans).to_excel('Q3_迭代.xlsx')
    ans = ans[:][1:]
    ans1 = []  # 记录坐标极径长度
    for i in ans:
        a = []
        for j in range(1, 10):
            a.append(i[j][0])
        ans1.append(a)
    plt.plot(np.std(np.array(ans1), axis=1), 'r')  # 画极径长度的标准差
    ans2 = []
    for i in ans:
        a = []
        for j in range(1, 10):
            a.append(math.fabs(i[j][1] - (j - 1) * 40))

        ans2.append(a)

    plt.plot(np.std(np.array(ans2), axis=1), 'b')  # 画和理想极角的差值的标准差
    plt.legend(['极径长度的标准差', '实际坐标极角和理想极角的差值的标准差'])
    plt.title('迭代次数与精度的变化图')
    plt.show()


def init():
    Polar = [(0, 0), (100, 0), (98, 40.10), (112, 80.21), (105, 119.75), (98, 159.86), (112, 199.96), (105, 240.07),
             (98, 280.17), (112, 320.28)]  # 初始化数据
    return Q3_solve(Polar)
```

```
if __name__ == "__main__":
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    init()
```