2022B

基于纯方位无源定位技术的无人机位置确定

摘要

针对问题一第一小问，建立了**基于极坐标的外接圆定位模型**，求得了被动接收信号无人机的定位位置。根据**圆周角定理**，判断出无人机运动轨迹是圆形。再根据三**角形的正弦定理**计算外接圆半径和求解圆的方程，并将外接圆方程联立得到极角，代入圆方程得到极径。针对如下两类特殊情形进行独立分析：当某一方位角为 0 时，说明两无人机方向相同，共线，可以直接获得角度，额外使用正弦定理求解具体位置；当公式中分母为零时，判断出无人机落在 y 轴上，此时极角为 $\pm\frac{\pi}{2}$，分两种情况得到具体位置。此外，根据不同 $\alpha_1, \alpha_2, \alpha_3$ 的大小和排列顺序，**分情况讨论**外接圆的构型与解的存在性，最终确定被动接收信号无人机的位置，建立得到定位模型。

针对问题一第二小问，建立了**单额外无人机角度比对模型**，求解了发射信号的无人机的编号、误差范围等问题。首先根据**对称性**，作出重要的假设确立：**接收信号的无人机知道自身编号**。并且利用发射信号的无人机编号未知这一特点，其位置只能在**预设离散点**中选取。通过对实际位置与理想位置接收到的方向信息的**比较**，分析得到，在一**定的误差范围内**，作表得到方向信息的**上下界**。对于得到的方向信息，可以查表获得发射信号的无人机的编号。随后，结合问题一第一小问中构建的**基于极坐标的外接圆定位模型**，以所识别发射源编号为基准，反推出目标接收无人机的几何位置，最终实现目标无人机的有效定位。

针对问题一第三小问，建立了**基于理想情况的模型说明**，求解了角度的具体调整方案。为使无人机收敛于某一半径的圆，需保证无人机与相邻无人机和中心位置无人机构成的夹角为 $\frac{7}{18}\pi$，构建成理想九边形。根据无人机当前位置与理想位置的偏差，分两种情况讨论：若某无人机位于目标圆外，则应向内侧调整；若位于圆内，则应向外侧调整。以当前偏离的角度为基准，通过角平分线方向调整其移动方向，确保无人机向理想位置靠拢。为避免调整过程中的震荡与跳变，引入**迭代更新策略**，设定每轮调整的步长，根据**环状序列**依次调整各无人机的位置。最终，构建出基于极坐标与迭代优化的队形收敛模型，使得整个无人机集群收敛到理想圆形结构，实现高精度的角度调整与稳定队形。

针对问题二，建立了**基于理想六边形的迭代定位模型**以及**基于理想菱形的迭代定位模型**，求解了角度的具体调整方案。锥形编队队形因为直线上相邻两架无人机的间距相等，将其拆分成六边形和菱形讨论，分解问题便于求解。对于理想六边形和菱形，采用类似问题一第三小问的思路，引入**迭代更新策略**，设定每轮调整的步长，不同之处在于调整各无人机的位置的顺序。

关键词：几何模型分析，方向角误差匹配，迭代队形调整

一、 问题重述

1.1 问题背景

为提升无人机编队在复杂电磁环境下的隐蔽性和抗干扰能力，任务执行时应尽可能保持电磁静默状态，即限制无线电信号的主动发射。然而，在保持电磁静默的前提下，编队中的各架无人机仍需维持协同一致的飞行状态与既定队形。

为此，任务提出采用“纯方位无源定位”方式进行编队控制。该方法中，仅有部分无人机发射信号，其余无人机通过被动接收方式感知方向信息，从而推算出自身在编队中的相对位置，实现位置修正。无人机间的通信不依赖距离测量，定位信息仅来源于方向夹角。

1.2 问题描述

1. **问题一第一小问**：9 架无人机均匀分布在某一圆周上，另一架位于圆心。通过几何模型分析，建立被动接收信号无人机的定位模型。
2. **问题一第二小问**：除编号为 $FY00$ 和 $FY01$ 的无人机发射信号，确定额外选择的编号未知的发射信号无人机数量，在误差范围内建立一种合适的待测无人机定位模型。
3. **问题一第三小问**：在初始位置存在偏差的情况下，通过多次选择 $FY00$ 和最多 3 架圆周上无人机发射信号，其余无人机接收方向信息并进行位置调整，建立迭代修正模型，实现无人机逐步收敛到均匀分布在某一圆周上的理想状态。
4. **问题二**：在问题一第三小问的基础上，将圆形编队改为锥形编队，建立新的纯方位无源定位下的无人机位置调整模型，设计合理的信号发射与位置修正策略，使无人机集群从初始偏差状态调整为等间距的理想队形。

二、 模型假设

- 假设 1：所有无人机均保持在同一水平面上飞行，可忽略高度差异的影响，问题可简化为二维平面几何问题。
- 假设 2：所有角度测量均为理想条件下的测量结果，不考虑测角误差、信号遮挡或噪声等因素的干扰。
- 假设 3：编队结构为刚性结构，在理想状态下各无人机相对位置固定，位置偏差仅在调整过程中出现，调整后可瞬时归位。
- 假设 4：接收信号的无人机知道自身编号。

三、 符号说明

符号	含义	单位
$FY0i$	第 i 架无人机的编号	/
$FY0j$	第 j 架无人机的编号	/
$FY0t$	待求无人机的编号	/
s	弧长	m
t	时间	s
θ_i	第 i 节板凳前把手中心的极角	°

四、 问题一（1）模型的建立与求解

针对问题一，为了确定位置略有偏差无人机的位置, 首先要以 FY00 为圆心建立极坐标系，再通过编队中另两架位置已知的无人机进行几何关系分析，进而得出可通过分别作外接圆求交点的方式得到位置略有偏差无人机的具体位置。对于具体方程的计算，可结合正弦定理等几何性质以及极坐标系下的圆坐标方程进行简化，最终可建立通过任意两个位置无偏差且编号已知无人机下被动接收信号无人机的定位模型。

4.1 问题一模型的构建

4.1.1 基于极坐标的外接圆定位模型

首先建立极坐标系，设原先无人机均匀分布圆周的半径为 R，第 i, j 架无人机的编号为 $FY0i, FU0j, (i, j = 1, 2...9$ 且 $i \neq j)$，对应点的坐标为 $FY0i(R, 40i), FY0j(R, 40j)$，位置偏差的点为 $FY0t(r_0, \theta_0)$，所测得的方位角为 α_1, α_2。分别以 $FYOi, FY00, FY0t$ 和 $FY0j, FY00, FY0t$ 为三角形作其外接圆如下图所示：

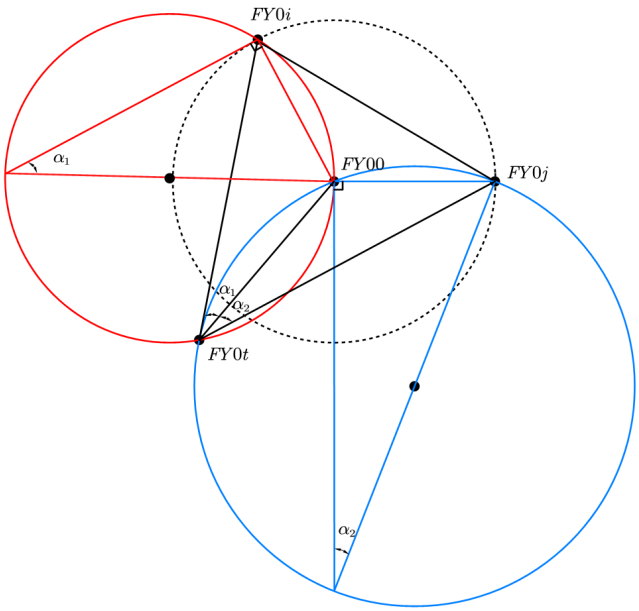


图 1　外接圆及交点示意图

由图示所示，两个外接圆可确定两个交点：一个为原点，另外一个为 $FY0t$, 因此只需要通过求解两外接圆的交点即可求解出位置偏差无人机的具体位置。为了化简计算，选取三角形 $FY00, FY0i, FY0t$ 进行分析，但由于无人机所在的位置不同，所确定的三角形形式也不同，因此总共可分为如下四种情况进行分析，并依据结果将最终外接圆圆心的极角分为两种情况，具体作图如下：

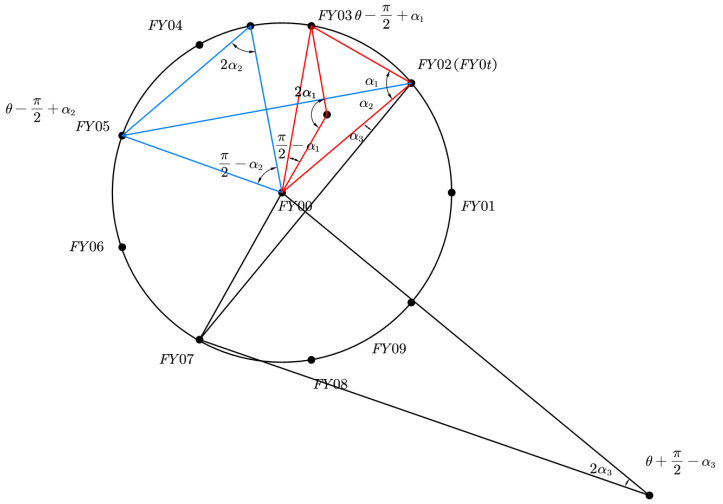


图 2　外接圆及交点示意图

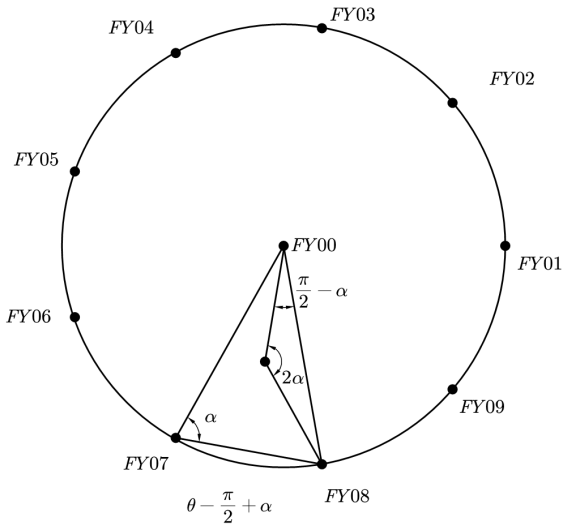


图 3　外接圆及交点示意图

(1) 当 $40(i-t)>\pi$ 时

由于 $FYOi, FY0j$ 与 x 轴构成的夹角分别为 $40i, 40j$, 根据几何性质可求出外接圆的圆心极角为:

$$\frac{\pi}{2}-\alpha_1+40(i-1) \tag{1}$$

根据圆的几何性质可求 $\angle OO_1FY0i=2\alpha_1$, 同时根据三角形的正弦定理可求该三角形的外接圆半径为:

$$r_1=\frac{R}{2\sin\alpha_1} \tag{2}$$

同理可求三角形 $FY00, FY0j, FY0t$ 的外接圆半径为:

$$r_2=\frac{R}{2\sin\alpha_2} \tag{3}$$

因此可确定圆 O_1 的圆心坐标为:

$$O_1\left(r_1,\theta_1\right)=\left(\frac{R}{2\sin\alpha_1},\frac{\pi}{2}-\alpha_1+40(i-1)\right) \tag{4}$$

根据圆心为 (ρ_0,θ_0)，半径为 r 圆的一般方程为:

$$\rho^2+\rho_0^2-2\rho\rho_0\cos\left(\theta-\theta_0\right)=r^2 \tag{5}$$

可求出圆 O_1 在极坐标系下的方程为:

$$r=2r_1\cos\left(\theta-\frac{\pi}{2}+\alpha_1-40(i-1)\right) \tag{6}$$

同理可求另一外接圆的方程为:

$$r=2r_2\cos\left(\theta-\frac{\pi}{2}+\alpha_2-40(j-1)\right) \tag{7}$$

联立两方程, 可最终求解出 $FY0t$ 的极角为:

$$\theta_0 = \arctan\left(-\frac{\cos\left(\alpha_2 - 40(j-1)\right)\sin\alpha_1 - \cos\left(\alpha_1 - 40(i-1)\right)\sin\alpha_2}{\sin\left(\alpha_2 - 40(j-1)\right)\sin\alpha_1 - \sin\left(\alpha_1 - 40(i-1)\right)\sin\alpha_2}\right) \tag{8}$$

将 θ_0 代入圆方程可确定点 $FY0t$ 的极径为:

$$r_0 = \sqrt{\frac{R}{\sin\alpha_1}\cos\left(\theta_0 - \frac{\pi}{2} + \alpha_1 - 40(i-1)\right)} \tag{9}$$

(2) 当 $40(i-t) < \pi$ 时由于 $FYOi, FY0j$ 与 x 轴构成的夹角分别为 $40i, 40j$, 根据几何性质可求出外接圆的圆心极角为:

$$\frac{-\pi}{2} + \alpha_1 + 40(i-1) \tag{10}$$

同理可求得此时两外接圆在极坐标下的方程为:

$$\begin{cases} r = 2r_1\cos\left(\theta + \dfrac{\pi}{2} - \alpha_1 - 40(i-1)\right) \\ r = 2r_2\cos\left(\theta + \dfrac{\pi}{2} - \alpha_2 - 40(j-1)\right) \end{cases} \tag{11}$$

联立求解出 $FY0t$ 的极角为:

$$\theta_0 = \arctan\left(-\frac{\cos\left(\alpha_2 - 40(j-1)\right)\sin\alpha_1 - \cos\left(\alpha_1 - 40(i-1)\right)\sin\alpha_2}{\sin\left(\alpha_2 - 40(j-1)\right)\sin\alpha_1 - \sin\left(\alpha_1 - 40(i-1)\right)\sin\alpha_2}\right) \tag{12}$$

将 θ_0 代入圆方程可确定点 $FY0t$ 的极径为:

$$r_0 = \sqrt{\frac{R}{\sin\alpha_1}\cos\left(\theta_0 + \frac{\pi}{2} - \alpha_1 - 40(i-1)\right)} \tag{13}$$

值得注意的是，以上的模型分析基于方位角 α_1, α_2 以及 θ_0 表达式分母不为 0 的情况，以下针对方位角中存在为零和分母为零的情况做出分析和解释。

当方位角为零时作图如下:

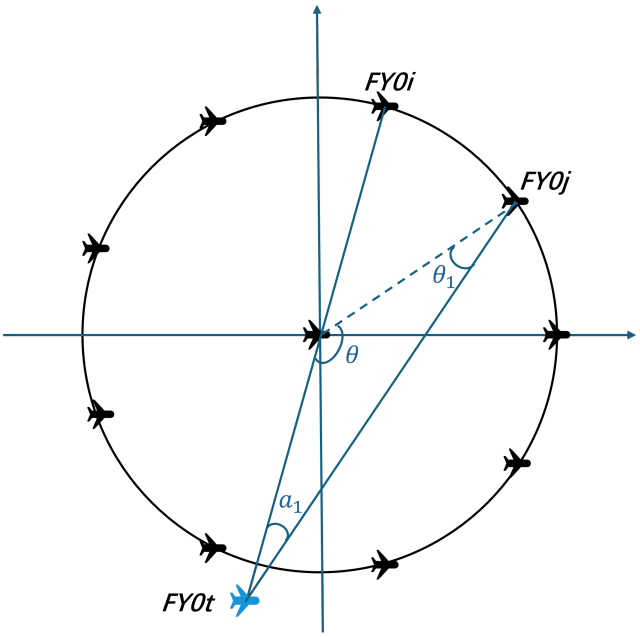


图 4　方位角为零时示意图

由图可知此时 $FY0t$ 位于一 $FY0i$ 与 FY00 的连线上，因此利用角度的关系可求得 $\theta_1 = 40\,(i-1) - 40\,(j-1) - \alpha_1$, 再在三角形中运用正弦定理有：

$$\frac{R}{\sin\alpha_1} = \frac{r_0}{\sin\theta_1} \tag{14}$$

由此可确定 $FY0t$ 的具体位置为：

$$(r_0, \theta_0) = \left(\frac{R\sin\left(40\,(i-1) - 40\,(j-1) - \alpha_1\right)}{\sin\alpha_1}, 40\,(i-1) + \pi\right) \tag{15}$$

当分母为零时：可知此时位置偏差的无人机落在了 y 轴上。因此在此基础上又可分为两种小情况

当 $\cos\left(\alpha_2 - 40(j-1)\right)\sin\alpha_1 - \cos\left(\alpha_1 - 40(i-1)\right)\sin\alpha_2) < 0$ 时，$FY0t$ 位于 y 轴上方，其具体位置为：

$$(r_0, \theta_0) = \left(\frac{\pi}{2}, 40\,(i-1) + \pi\right) \tag{16}$$

当 $\cos\left(\alpha_2 - 40(j-1)\right)\sin\alpha_1 - \cos\left(\alpha_1 - 40(i-1)\right)\sin\alpha_2) > 0$ 时，$FY0t$ 位于 y 轴下方，$FY0t$ 的具体位置为：

$$(r_0, \theta_0) = \left(\frac{-\pi}{2}, 40\,(i-1) + \pi\right) \tag{17}$$

五、 问题一（2）模型的建立与求解

针对问题一的第二问，除 FY00 和 FY01 发射信号外，需要判断额外需要的编号未知的无人机的数量，来实现位置略有偏差的无人机的有效定位。

5.1 假设确立

这里假设接收信号的无人机知道自己的编号，否则根据图形的对称性难以判断无人机的具体编号，如下图：

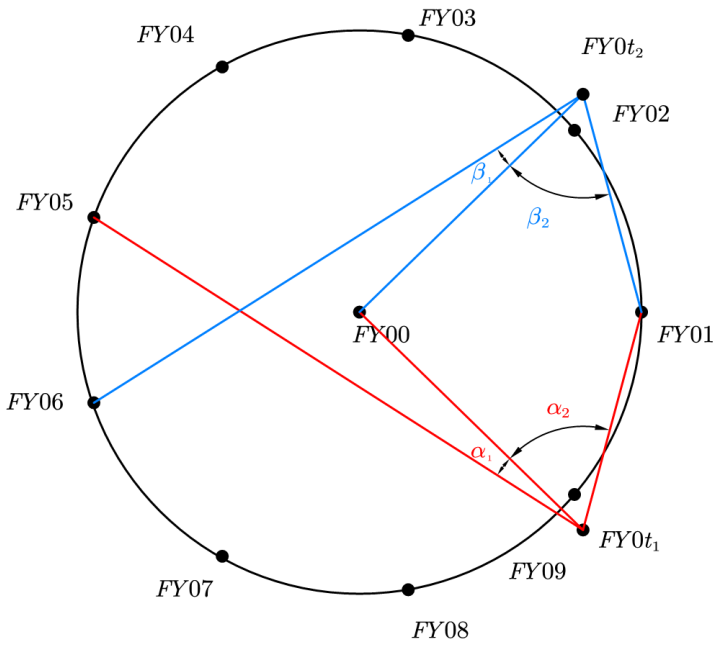


图 5　对称性示意图

对于两种情况，其中 $FY0t_1$ 与 $FY0t_2$ 关于 $FY00$ 和 $FY01$ 形成的圆的直径上下对称，可见有 $\alpha_1 = \beta_1$，$\alpha_2 = \beta_2$，在这种情况下角度相同，因而无法判断接收信号的无人机的自身编号。

而对于多个无人机，情况类似，这里不再赘述。

虽然可以通过后续的分析判断出无人机属于两个编号中的一个，但无法精确确定无人机位置。并且接收信号的无人机知道自己的编号符合现实实际中调整无人机位置的要求，故给出假设：接收信号的无人机知道自己的编号，设该编号为 $FY0t$，且 t 已知。

5.2 未知编号发射信号的无人机的编号确定

设未知编号发射信号的无人机的编号为 $FY0i$，这里令 $i = 2, 3$, 画出一个额外无人机情况下的示意图如下：

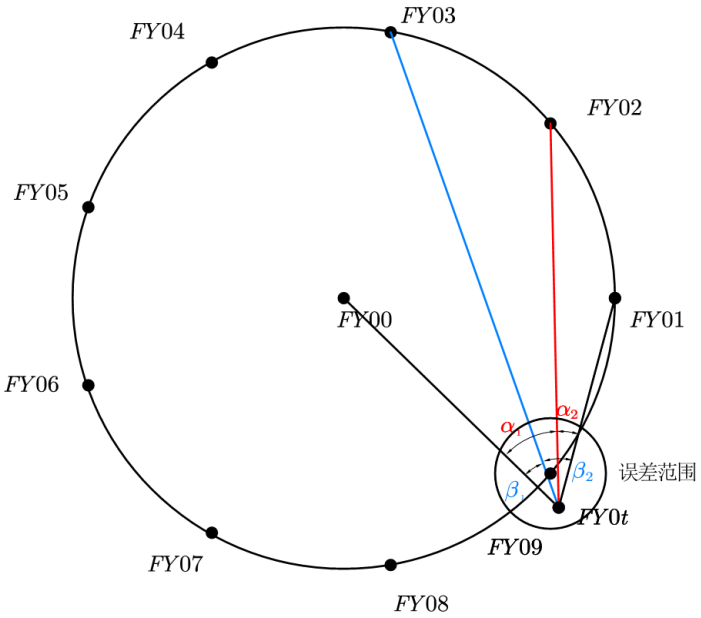


图 6　未知编号发射信号的无人机位于不同位置的示意图

由图中可以看出对于未知编号发射信号的无人机，不同的编号形成的角度是不同的。令其在 $FY02$，会形成角 α_1、α_2；在 $FY03$，会形成角 β_1、β_2。并且，无人机的分布不是连续性的，而是离散性的，其极角必是 $40°$ 的整数倍。

因此，在待求无人机的编号为 $FY09$，即实际位置在点 $FY09$ 附近时，我们可以通过接收的角度与理想位置形成的角度大小的比较判断来确定未知编号发射信号的无人机的编号。依此类推，对于每一个不同编号的接收信号的无人机，都可以通过该方法确定未知发射无人机的编号。

并且，因为无人机位置略有偏差，依据现实社会经验，我们假设无人机的误差范围为以理想位置为圆心的一个圆。在偏差较小的时候可以通过角度准确判断位置发射信号的无人机的编号，在偏差大大到一定程度时，则无法精确判断发射信号的无人机的编号，如下图：

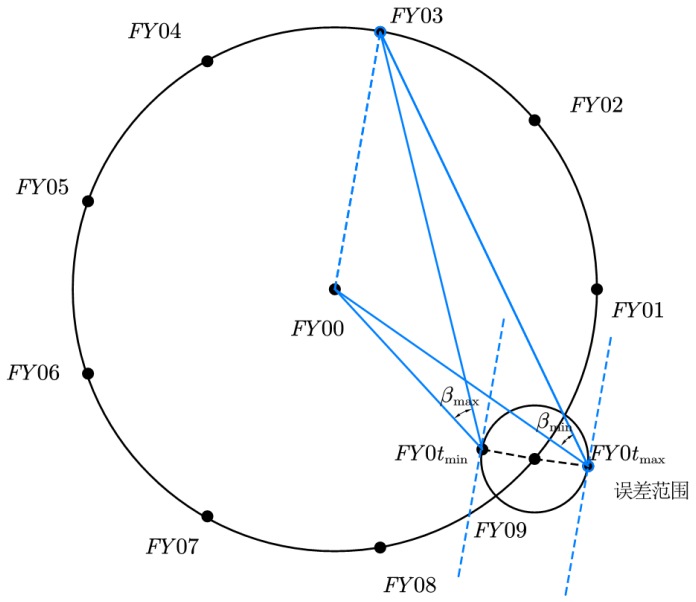


图 7　单个无人机（FY03）偏差示意图

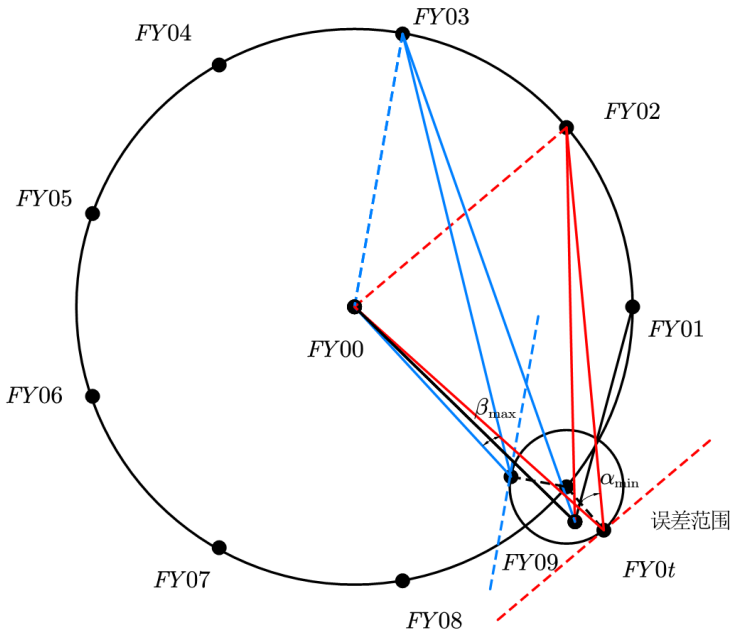


图 8　两个无人机（FY03、FY02）偏差示意图

对于单个无人机，当 $FY0t$ 落在 $FY09$ 的误差范围内时，可通过几何关系作出两条切线，得到 β_{max} 和 β_{min}，即 $\beta \in [\beta_{\min}, \beta_{\max}]$。其中 β 是无人机方向信息，β_{max} 表示在误差范围内无人机接收到方向信息 β 可能取到的最大值，β_{min} 表示在误差范围内无人机接收到方向信息 β 可能取到的最小值。

对于两个无人机，同理 $\alpha \in [\alpha_{\min}, \alpha_{\max}]$,$\beta \in [\beta_{\min}, \beta_{\max}]$。在理想位置上的 $\alpha_0 > \beta_0$。若有 $\alpha_{max} > \alpha_{min} > \beta_{max} > \beta_{min}$，则可以正确判断未知编号发射信号无人机的编号。理想位置上 $\alpha_{max} = \alpha_{min}$，$\beta_{max} = \beta_{min}$，该式恒成立。

如果 $\alpha_{\min} < \beta_{\max}$，这说明范围发生重合，不能通过角度大小比较，来判断未知编号发射信号无人机的编号。

现给出形成角度的表格，用于判断编号，如下表 (其中 T 为待测点无人机编号，I/J 为发射信号的无人机编号)：

表 1　角度数据表

(a) $\angle FY00 - FY0T - FY0I$ **(单位：度)**

T	I	角度表示	角度值
2	1	$\angle FY00 - FY02 - FY01$	70.0000°
2	3	$\angle FY00 - FY02 - FY03$	70.0000°
2	4	$\angle FY00 - FY02 - FY04$	50.0000°
2	5	$\angle FY00 - FY02 - FY05$	30.0000°
2	6	$\angle FY00 - FY02 - FY06$	10.0000°
2	7	$\angle FY00 - FY02 - FY07$	10.0000°
2	8	$\angle FY00 - FY02 - FY08$	30.0000°
2	9	$\angle FY00 - FY02 - FY09$	50.0000°
3	1	$\angle FY00 - FY03 - FY01$	50.0000°
3	2	$\angle FY00 - FY03 - FY02$	70.0000°
3	4	$\angle FY00 - FY03 - FY04$	70.0000°
3	5	$\angle FY00 - FY03 - FY05$	50.0000°
3	6	$\angle FY00 - FY03 - FY06$	30.0000°
3	7	$\angle FY00 - FY03 - FY07$	10.0000°
3	8	$\angle FY00 - FY03 - FY08$	10.0000°
3	9	$\angle FY00 - FY03 - FY09$	30.0000°
4	1	$\angle FY00 - FY04 - FY01$	30.0000°
4	2	$\angle FY00 - FY04 - FY02$	50.0000°
4	3	$\angle FY00 - FY04 - FY03$	70.0000°
4	5	$\angle FY00 - FY04 - FY05$	70.0000°
4	6	$\angle FY00 - FY04 - FY06$	50.0000°
4	7	$\angle FY00 - FY04 - FY07$	30.0000°
4	8	$\angle FY00 - FY04 - FY08$	10.0000°
4	9	$\angle FY00 - FY04 - FY09$	10.0000°
5	1	$\angle FY00 - FY05 - FY01$	10.0000°
5	2	$\angle FY00 - FY05 - FY02$	30.0000°
5	3	$\angle FY00 - FY05 - FY03$	50.0000°
5	4	$\angle FY00 - FY05 - FY04$	70.0000°
5	6	$\angle FY00 - FY05 - FY06$	70.0000°
5	7	$\angle FY00 - FY05 - FY07$	50.0000°
5	8	$\angle FY00 - FY05 - FY08$	30.0000°
5	9	$\angle FY00 - FY05 - FY09$	10.0000°

(b) $\angle FY00 - FY0T - FY0I$ **(单位：度)**

T	I	角度表示	角度值
6	1	$\angle FY00 - FY06 - FY01$	10.0000°
6	2	$\angle FY00 - FY06 - FY02$	10.0000°
6	3	$\angle FY00 - FY06 - FY03$	30.0000°
6	4	$\angle FY00 - FY06 - FY04$	50.0000°
6	5	$\angle FY00 - FY06 - FY05$	70.0000°
6	7	$\angle FY00 - FY06 - FY07$	70.0000°
6	8	$\angle FY00 - FY06 - FY08$	50.0000°
6	9	$\angle FY00 - FY06 - FY09$	30.0000°
7	1	$\angle FY00 - FY07 - FY01$	30.0000°
7	2	$\angle FY00 - FY07 - FY02$	10.0000°
7	3	$\angle FY00 - FY07 - FY03$	10.0000°
7	4	$\angle FY00 - FY07 - FY04$	30.0000°
7	5	$\angle FY00 - FY07 - FY05$	50.0000°
7	6	$\angle FY00 - FY07 - FY06$	70.0000°
7	8	$\angle FY00 - FY07 - FY08$	70.0000°
7	9	$\angle FY00 - FY07 - FY09$	50.0000°
8	1	$\angle FY00 - FY08 - FY01$	50.0000°
8	2	$\angle FY00 - FY08 - FY02$	30.0000°
8	3	$\angle FY00 - FY08 - FY03$	10.0000°
8	4	$\angle FY00 - FY08 - FY04$	10.0000°
8	5	$\angle FY00 - FY08 - FY05$	30.0000°
8	6	$\angle FY00 - FY08 - FY06$	50.0000°
8	7	$\angle FY00 - FY08 - FY07$	70.0000°
8	9	$\angle FY00 - FY08 - FY09$	70.0000°
9	1	$\angle FY00 - FY09 - FY01$	70.0000°
9	2	$\angle FY00 - FY09 - FY02$	50.0000°
9	3	$\angle FY00 - FY09 - FY03$	30.0000°
9	4	$\angle FY00 - FY09 - FY04$	10.0000°
9	5	$\angle FY00 - FY09 - FY05$	10.0000°
9	6	$\angle FY00 - FY09 - FY06$	30.0000°
9	7	$\angle FY00 - FY09 - FY07$	50.0000°
9	8	$\angle FY00 - FY09 - FY08$	70.0000°

表 2　角度数据表

(a) $\angle FY00-FY0T-FY0I$ **(单位：度)**

T	I	角度表示	角度值
2	3	$\angle FY02-FY03-FY01$	70.0000°
2	4	$\angle FY02-FY04-FY01$	50.0000°
2	5	$\angle FY02-FY05-FY01$	30.0000°
2	6	$\angle FY02-FY06-FY01$	10.0000°
2	7	$\angle FY02-FY07-FY01$	10.0000°
2	8	$\angle FY02-FY08-FY01$	30.0000°
2	9	$\angle FY02-FY09-FY01$	50.0000°
3	2	$\angle FY03-FY02-FY01$	70.0000°
3	4	$\angle FY03-FY04-FY01$	70.0000°
3	5	$\angle FY03-FY05-FY01$	50.0000°
3	6	$\angle FY03-FY06-FY01$	30.0000°
3	7	$\angle FY03-FY07-FY01$	10.0000°
3	8	$\angle FY03-FY08-FY01$	10.0000°
3	9	$\angle FY03-FY09-FY01$	30.0000°
4	2	$\angle FY04-FY02-FY01$	50.0000°
4	3	$\angle FY04-FY03-FY01$	70.0000°
4	5	$\angle FY04-FY05-FY01$	70.0000°
4	6	$\angle FY04-FY06-FY01$	50.0000°
4	7	$\angle FY04-FY07-FY01$	30.0000°
4	8	$\angle FY04-FY08-FY01$	10.0000°
4	9	$\angle FY04-FY09-FY01$	10.0000°
5	2	$\angle FY05-FY02-FY01$	30.0000°
5	3	$\angle FY05-FY03-FY01$	50.0000°
5	4	$\angle FY05-FY04-FY01$	70.0000°
5	6	$\angle FY05-FY06-FY01$	70.0000°
5	7	$\angle FY05-FY07-FY01$	50.0000°
5	8	$\angle FY05-FY08-FY01$	30.0000°
5	9	$\angle FY05-FY09-FY01$	10.0000°

(b) $\angle FY00-FY0T-FY0I$ **(单位：度)**

T	I	角度表示	角度值
6	2	$\angle FY06-FY02-FY01$	10.0000°
6	3	$\angle FY06-FY03-FY01$	30.0000°
6	4	$\angle FY06-FY04-FY01$	50.0000°
6	5	$\angle FY06-FY05-FY01$	70.0000°
6	7	$\angle FY06-FY07-FY01$	70.0000°
6	8	$\angle FY06-FY08-FY01$	50.0000°
6	9	$\angle FY06-FY09-FY01$	30.0000°
7	2	$\angle FY07-FY02-FY01$	10.0000°
7	3	$\angle FY07-FY03-FY01$	10.0000°
7	4	$\angle FY07-FY04-FY01$	30.0000°
7	5	$\angle FY07-FY05-FY01$	50.0000°
7	6	$\angle FY07-FY06-FY01$	70.0000°
7	8	$\angle FY07-FY08-FY01$	70.0000°
7	9	$\angle FY07-FY09-FY01$	50.0000°
8	2	$\angle FY08-FY02-FY01$	30.0000°
8	3	$\angle FY08-FY03-FY01$	10.0000°
8	4	$\angle FY08-FY04-FY01$	10.0000°
8	5	$\angle FY08-FY05-FY01$	30.0000°
8	6	$\angle FY08-FY06-FY01$	50.0000°
8	7	$\angle FY08-FY07-FY01$	70.0000°
8	9	$\angle FY08-FY09-FY01$	70.0000°
9	2	$\angle FY09-FY02-FY01$	50.0000°
9	3	$\angle FY09-FY03-FY01$	30.0000°
9	4	$\angle FY09-FY04-FY01$	10.0000°
9	5	$\angle FY09-FY05-FY01$	10.0000°
9	6	$\angle FY09-FY06-FY01$	30.0000°
9	7	$\angle FY09-FY07-FY01$	50.0000°
9	8	$\angle FY09-FY08-FY01$	70.0000°

根据问题一的第三小问的数据，当误差范围为 $|\Delta R|<15$ 时，只需将角度修改上下界即可通过判断是否在范围内，进而精确判断发射信号的无人机的编号，如下表：

表 3　角度数据表

(a) $\angle FY00-FY0T-FY0I$ **(单位：度)**

T	I	角度表示	角度范围
2	1	$\angle FY00-FY02-FY01$	[58.9833°, 84.5978°]
2	3	$\angle FY00-FY02-FY03$	[58.9833°, 84.5978°]
2	4	$\angle FY00-FY02-FY04$	[44.0004°, 57.5533°]
2	5	$\angle FY00-FY02-FY05$	[25.3517°, 35.4011°]
2	6	$\angle FY00-FY02-FY06$	[5.6975°, 14.5324°]
2	7	$\angle FY00-FY02-FY07$	[5.6975°, 14.5324°]
2	8	$\angle FY00-FY02-FY08$	[25.3517°, 35.4011°]
2	9	$\angle FY00-FY02-FY09$	[44.0004°, 57.5533°]
3	1	$\angle FY00-FY03-FY01$	[44.0004°, 57.5533°]
3	2	$\angle FY00-FY03-FY02$	[58.9833°, 84.5978°]
3	4	$\angle FY00-FY03-FY04$	[58.9833°, 84.5978°]
3	5	$\angle FY00-FY03-FY05$	[44.0004°, 57.5533°]
3	6	$\angle FY00-FY03-FY06$	[25.3517°, 35.4011°]
3	7	$\angle FY00-FY03-FY07$	[5.6975°, 14.5324°]
3	8	$\angle FY00-FY03-FY08$	[5.6975°, 14.5324°]

(b) $\angle FY00-FY0T-FY0I$ **(单位：度)**

T	I	角度表示	角度范围
6	1	$\angle FY00-FY06-FY01$	[5.6975°, 14.5324°]
6	2	$\angle FY00-FY06-FY02$	[5.6975°, 14.5324°]
6	3	$\angle FY00-FY06-FY03$	[25.3517°, 35.4011°]
6	4	$\angle FY00-FY06-FY04$	[44.0004°, 57.5533°]
6	5	$\angle FY00-FY06-FY05$	[58.9833°, 84.5978°]
6	7	$\angle FY00-FY06-FY07$	[58.9833°, 84.5978°]
6	8	$\angle FY00-FY06-FY08$	[44.0004°, 57.5533°]
6	9	$\angle FY00-FY06-FY09$	[25.3517°, 35.4011°]
7	1	$\angle FY00-FY07-FY01$	[25.3517°, 35.4011°]
7	2	$\angle FY00-FY07-FY02$	[5.6975°, 14.5324°]
7	3	$\angle FY00-FY07-FY03$	[5.6975°, 14.5324°]
7	4	$\angle FY00-FY07-FY04$	[25.3517°, 35.4011°]
7	5	$\angle FY00-FY07-FY05$	[44.0004°, 57.5533°]
7	6	$\angle FY00-FY07-FY06$	[58.9833°, 84.5978°]
7	8	$\angle FY00-FY07-FY08$	[58.9833°, 84.5978°]

表 4　角度数据表 (续)

(a) $\angle FY00-FY0T-FY0I$ **(单位：度)**

T	I	角度表示	角度范围
3	9	$\angle FY00-FY03-FY09$	[25.3517°, 35.4011°]
4	1	$\angle FY00-FY04-FY01$	[25.3517°, 35.4011°]
4	2	$\angle FY00-FY04-FY02$	[44.0004°, 57.5533°]
4	3	$\angle FY00-FY04-FY03$	[58.9833°, 84.5978°]
4	5	$\angle FY00-FY04-FY05$	[58.9833°, 84.5978°]
4	6	$\angle FY00-FY04-FY06$	[44.0004°, 57.5533°]
4	7	$\angle FY00-FY04-FY07$	[25.3517°, 35.4011°]
4	8	$\angle FY00-FY04-FY08$	[5.6975°, 14.5324°]
4	9	$\angle FY00-FY04-FY09$	[5.6975°, 14.5324°]
5	1	$\angle FY00-FY05-FY01$	[5.6975°, 14.5324°]
5	2	$\angle FY00-FY05-FY02$	[25.3517°, 35.4011°]
5	3	$\angle FY00-FY05-FY03$	[44.0004°, 57.5533°]
5	4	$\angle FY00-FY05-FY04$	[58.9833°, 84.5978°]
5	6	$\angle FY00-FY05-FY06$	[58.9833°, 84.5978°]
5	7	$\angle FY00-FY05-FY07$	[44.0004°, 57.5533°]
5	8	$\angle FY00-FY05-FY08$	[25.3517°, 35.4011°]
5	9	$\angle FY00-FY05-FY09$	[5.6975°, 14.5324°]

(b) $\angle FY00-FY0T-FY0I$ **(单位：度)**

T	I	角度表示	角度范围
7	9	$\angle FY00-FY07-FY09$	[44.0004°, 57.5533°]
8	1	$\angle FY00-FY08-FY01$	[44.0004°, 57.5533°]
8	2	$\angle FY00-FY08-FY02$	[25.3517°, 35.4011°]
8	3	$\angle FY00-FY08-FY03$	[5.6975°, 14.5324°]
8	4	$\angle FY00-FY08-FY04$	[5.6975°, 14.5324°]
8	5	$\angle FY00-FY08-FY05$	[25.3517°, 35.4011°]
8	6	$\angle FY00-FY08-FY06$	[44.0004°, 57.5533°]
8	7	$\angle FY00-FY08-FY07$	[58.9833°, 84.5978°]
8	9	$\angle FY00-FY08-FY09$	[58.9833°, 84.5978°]
9	1	$\angle FY00-FY09-FY01$	[58.9833°, 84.5978°]
9	2	$\angle FY00-FY09-FY02$	[44.0004°, 57.5533°]
9	3	$\angle FY00-FY09-FY03$	[25.3517°, 35.4011°]
9	4	$\angle FY00-FY09-FY04$	[5.6975°, 14.5324°]
9	5	$\angle FY00-FY09-FY05$	[5.6975°, 14.5324°]
9	6	$\angle FY00-FY09-FY06$	[25.3517°, 35.4011°]
9	7	$\angle FY00-FY09-FY07$	[44.0004°, 57.5533°]
9	8	$\angle FY00-FY09-FY08$	[58.9833°, 84.5978°]

表 5　角度数据表

(a) $\angle FY0J\text{-}FY0T\text{-}FY01$ **(度)**

J	T	角度表示	角度范围
2	3	$\angle FY02\text{-}FY03\text{-}FY01$	[58.9833°, 84.5978°]
2	4	$\angle FY02\text{-}FY04\text{-}FY01$	[44.0004°, 57.5533°]
2	5	$\angle FY02\text{-}FY05\text{-}FY01$	[25.3517°, 35.4011°]
2	6	$\angle FY02\text{-}FY06\text{-}FY01$	[5.6975°, 14.5324°]
2	7	$\angle FY02\text{-}FY07\text{-}FY01$	[5.6975°, 14.5324°]
2	8	$\angle FY02\text{-}FY08\text{-}FY01$	[25.3517°, 35.4011°]
2	9	$\angle FY02\text{-}FY09\text{-}FY01$	[44.0004°, 57.5533°]
3	2	$\angle FY03\text{-}FY02\text{-}FY01$	[58.9833°, 84.5978°]
3	4	$\angle FY03\text{-}FY04\text{-}FY01$	[58.9833°, 84.5978°]
3	5	$\angle FY03\text{-}FY05\text{-}FY01$	[44.0004°, 57.5533°]
3	6	$\angle FY03\text{-}FY06\text{-}FY01$	[25.3517°, 35.4011°]
3	7	$\angle FY03\text{-}FY07\text{-}FY01$	[5.6975°, 14.5324°]
3	8	$\angle FY03\text{-}FY08\text{-}FY01$	[5.6975°, 14.5324°]
3	9	$\angle FY03\text{-}FY09\text{-}FY01$	[25.3517°, 35.4011°]
4	2	$\angle FY04\text{-}FY02\text{-}FY01$	[44.0004°, 57.5533°]
4	3	$\angle FY04\text{-}FY03\text{-}FY01$	[58.9833°, 84.5978°]
4	5	$\angle FY04\text{-}FY05\text{-}FY01$	[58.9833°, 84.5978°]
4	6	$\angle FY04\text{-}FY06\text{-}FY01$	[44.0004°, 57.5533°]
4	7	$\angle FY04\text{-}FY07\text{-}FY01$	[25.3517°, 35.4011°]
4	8	$\angle FY04\text{-}FY08\text{-}FY01$	[5.6975°, 14.5324°]
4	9	$\angle FY04\text{-}FY09\text{-}FY01$	[5.6975°, 14.5324°]
5	2	$\angle FY05\text{-}FY02\text{-}FY01$	[25.3517°, 35.4011°]
5	3	$\angle FY05\text{-}FY03\text{-}FY01$	[44.0004°, 57.5533°]
5	4	$\angle FY05\text{-}FY04\text{-}FY01$	[58.9833°, 84.5978°]
5	6	$\angle FY05\text{-}FY06\text{-}FY01$	[58.9833°, 84.5978°]
5	7	$\angle FY05\text{-}FY07\text{-}FY01$	[44.0004°, 57.5533°]
5	8	$\angle FY05\text{-}FY08\text{-}FY01$	[25.3517°, 35.4011°]
5	9	$\angle FY05\text{-}FY09\text{-}FY01$	[5.6975°, 14.5324°]

(b) $\angle FY0J\text{-}FY0T\text{-}FY01$ **(度)**

J	T	角度表示	角度范围
6	2	$\angle FY06\text{-}FY02\text{-}FY01$	[5.6975°, 14.5324°]
6	3	$\angle FY06\text{-}FY03\text{-}FY01$	[25.3517°, 35.4011°]
6	4	$\angle FY06\text{-}FY04\text{-}FY01$	[44.0004°, 57.5533°]
6	5	$\angle FY06\text{-}FY05\text{-}FY01$	[58.9833°, 84.5978°]
6	7	$\angle FY06\text{-}FY07\text{-}FY01$	[58.9833°, 84.5978°]
6	8	$\angle FY06\text{-}FY08\text{-}FY01$	[44.0004°, 57.5533°]
6	9	$\angle FY06\text{-}FY09\text{-}FY01$	[25.3517°, 35.4011°]
7	2	$\angle FY07\text{-}FY02\text{-}FY01$	[5.6975°, 14.5324°]
7	3	$\angle FY07\text{-}FY03\text{-}FY01$	[5.6975°, 14.5324°]
7	4	$\angle FY07\text{-}FY04\text{-}FY01$	[25.3517°, 35.4011°]
7	5	$\angle FY07\text{-}FY05\text{-}FY01$	[44.0004°, 57.5533°]
7	6	$\angle FY07\text{-}FY06\text{-}FY01$	[58.9833°, 84.5978°]
7	8	$\angle FY07\text{-}FY08\text{-}FY01$	[58.9833°, 84.5978°]
7	9	$\angle FY07\text{-}FY09\text{-}FY01$	[44.0004°, 57.5533°]
8	2	$\angle FY08\text{-}FY02\text{-}FY01$	[25.3517°, 35.4011°]
8	3	$\angle FY08\text{-}FY03\text{-}FY01$	[5.6975°, 14.5324°]
8	4	$\angle FY08\text{-}FY04\text{-}FY01$	[5.6975°, 14.5324°]
8	5	$\angle FY08\text{-}FY05\text{-}FY01$	[25.3517°, 35.4011°]
8	6	$\angle FY08\text{-}FY06\text{-}FY01$	[44.0004°, 57.5533°]
8	7	$\angle FY08\text{-}FY07\text{-}FY01$	[58.9833°, 84.5978°]
8	9	$\angle FY08\text{-}FY09\text{-}FY01$	[58.9833°, 84.5978°]
9	2	$\angle FY09\text{-}FY02\text{-}FY01$	[44.0004°, 57.5533°]
9	3	$\angle FY09\text{-}FY03\text{-}FY01$	[25.3517°, 35.4011°]
9	4	$\angle FY09\text{-}FY04\text{-}FY01$	[5.6975°, 14.5324°]
9	5	$\angle FY09\text{-}FY05\text{-}FY01$	[5.6975°, 14.5324°]
9	6	$\angle FY09\text{-}FY06\text{-}FY01$	[25.3517°, 35.4011°]
9	7	$\angle FY09\text{-}FY07\text{-}FY01$	[44.0004°, 57.5533°]
9	8	$\angle FY09\text{-}FY08\text{-}FY01$	[58.9833°, 84.5978°]

5.3 已知三无人机定位模型

获得角度 α_1，α_2 查表得到发射信号的无人机的编号 $FY0m$ 后，即代入问题一的第一小问的算法，实现对待测无人机的有效实际位置定位。

所以，依问题一的第三小问所示，在较小的误差范围下，只需一个额外的无人机就

能实现待测无人机的有效定位。

六、 问题一（3）模型的建立与求解

针对问题一 (3)，首先分析了理想状况下待测无人机与相邻两个无人机的夹角情况，从而对建议情况下不同测量方位角时移动方向进行了说明。针对实际中具体移动方向以及移动的距离大小，可结合角平分线确定方向，并用实际与理想夹角的差值确定具体的大小，从而确定每一次移动的具体决策。再通过人为设定循环次数以及调整步长，可构建出基于理想圆的迭代模型，从而所有无人机位置略有偏差时的定位模型和调整方案。

6.1 问题一（3）模型的建立

6.1.1 基于理想情况的模型说明

可知在理想情况下，一架无人机位于圆心, 另九架无人机均匀分布一定半径的圆周上，且所有无人机位置不发生偏差，此时构成一个正九边形，则无人机与相邻无人机和中心位置无人机构成的夹角为 $\frac{7}{18}\pi$, 如下图所示：

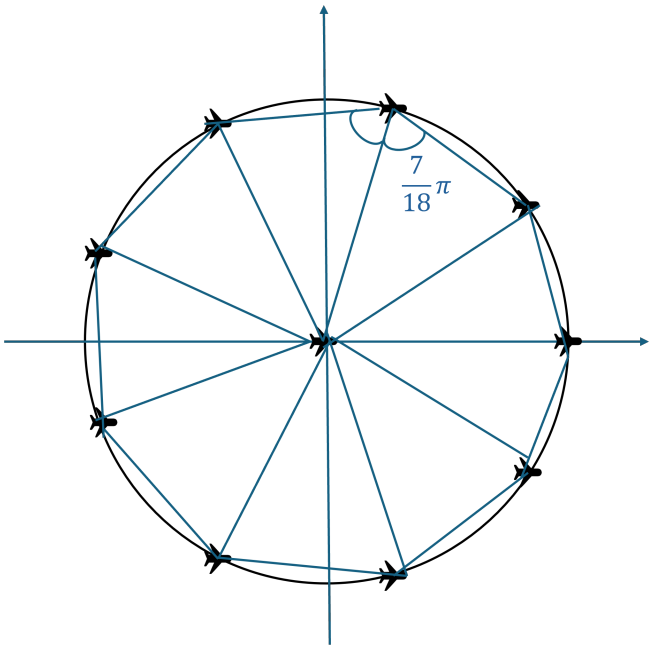


图 9　理想九边形示意图

根据理想九边形，无人机的位置偏差可能导致位置在九边形之外也可能在其内部，因此可以对无人机位置出现偏差的两种简易情况进行分析，

两种情况分别如下图所示：

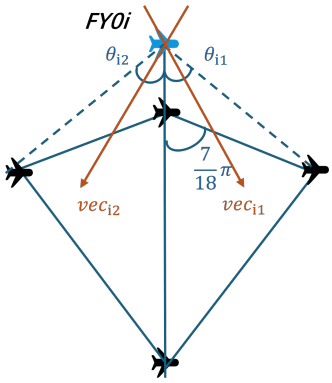


图 10　无人机在九边形外

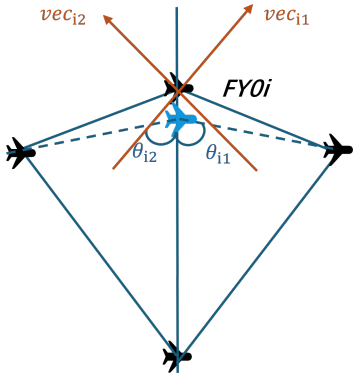


图 11　无人机在九边形内

由图可知，当无人机在九边形外部时，显然有 $\theta_{i2}, \theta_i 1$ 均小于 $\frac{7}{18}\pi$，因此，可分别取向内的角平分线确定方向向量 vec_{i1}, vec_{i2}, 从而可依据两方向向量的和确定无人机此次调整的方向, 调整使得 $\theta_{i2}, \theta_i 1$ 变大，往 $\frac{7}{18}\pi$ 趋近。

当无人机在九边形内部时，显然有 $\theta_{i2}, \theta_i 1$ 均大于 $\frac{7}{18}\pi$，因此，可分别取向外的角平分线确定方向向量 vec_{i1}, vec_{i2}, 从而可依据两方向向量的和确定无人机此次调整的方向，调整使得 $\theta_{i2}, \theta_i 1$ 变小，往 $\frac{7}{18}\pi$ 趋近。

6.1.2 基于理想九边形的迭代定位模型

为了确定方向向量 vec_{i1}, vec_{i2} 的具体方位，设 $FY0i$ 的极角为 θ_i, 并分为以下几种情况进行讨论：

(一) 当无人机位于 x 轴上方时：

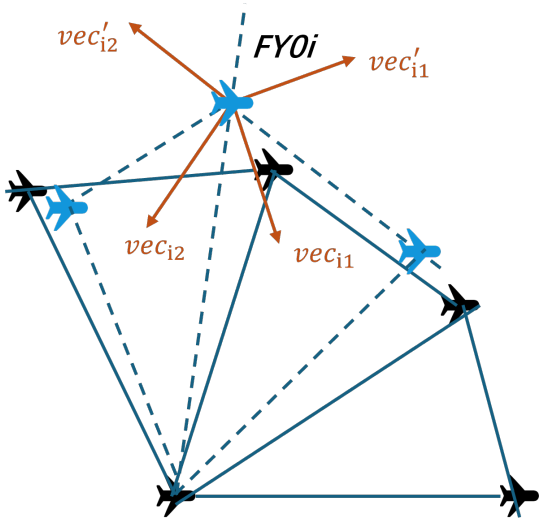


图 12　无人机位于 x 轴上方示意图

若 $\theta_{i1} < \frac{7}{18}\pi$, 则有

$$\theta_{\text{veci1}} = \theta_i + \frac{\theta_{i1}}{2} + \pi \tag{18}$$

$$\vec{vec}_{i1} = \left(\cos\left(\theta_{vec_{i1}}\right), \sin\left(\theta_{vec_{i1}}\right)\right) \tag{19}$$

若 $\theta_{i1} > \frac{7}{18}\pi$, 则有

$$\theta'_{veci1} = \theta_{veci1} + \frac{\pi}{2} \tag{20}$$

$$\vec{vec}'_{i1} = \left(\cos\left(\theta'_{vec_{i1}}\right), \sin\left(\theta'_{vec_{i1}}\right)\right) \tag{21}$$

若 $\theta_{i2} < \frac{7}{18}\pi$, 则有

$$\theta_{\text{veci2}} = \theta_i - \frac{\theta_{i2}}{2} + \pi \tag{22}$$

$$\vec{vec}_{i2} = \left(\cos\left(\theta_{vec_{i2}}\right), \sin\left(\theta_{vec_{i2}}\right)\right) \tag{23}$$

若 $\theta_{i2} > \frac{7}{18}\pi$, 则有

$$\theta'_{veci1} = \theta_{veci1} - \frac{\pi}{2} \tag{24}$$

$$\vec{vec}'_{i2} = \left(\cos\left(\theta'_{vec_{i2}}\right), \sin\left(\theta'_{vec_{i2}}\right)\right) \tag{25}$$

(二) 当无人机位于 x 轴下方时:

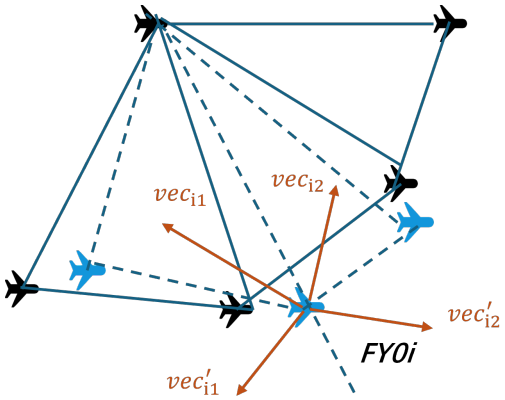


图 13　无人机位于 x 轴下方示意图

若 $\theta_{i1} < \frac{7}{18}\pi$, 则有

$$\theta_{\text{veci1}} = \frac{\theta_{i1}}{2} - \theta_i - \pi \tag{26}$$

$$\vec{vec}_{i1} = \left(\cos\left(\theta_{vec_{i1}}\right), \sin\left(\theta_{vec_{i1}}\right)\right) \tag{27}$$

若 $\theta_{i1} > \frac{7}{18}\pi$, 则有

$$\theta'_{veci1} = \theta_{veci1} + \frac{\pi}{2} \tag{28}$$

$$\vec{vec}'_{i1} = \left(\cos\left(\theta'_{vec_{i1}}\right), \sin\left(\theta'_{vec_{i1}}\right)\right) \tag{29}$$

若 $\theta_{i2} < \frac{7}{18}\pi$, 则有

$$\theta_{\text{veci2}} = -\frac{\theta_{i2}}{2} - \theta_i - \pi \tag{30}$$

$$\vec{vec}_{i2} = (\cos(\theta_{vec_{i2}}), \sin(\theta_{vec_{i2}})) \tag{31}$$

若 $\theta_{i2} > \frac{7}{18}\pi$, 则有

$$\theta'_{veci2} = \theta_{veci2} - \frac{\pi}{2} \tag{32}$$

$$\vec{vec}'_{i2} = \left(\cos\left(\theta'_{vec_{i2}}\right), \sin\left(\theta'_{vec_{i2}}\right)\right) \tag{33}$$

根据以上的讨论，可得到不同情况下角平分线的方向向量值，从而初步确定了无人机的调整方向。为了使得调整的方向更为精确，可通过计算 θ_i1, θ_i2 与 $\frac{7}{18}\pi$ 的差值，来确定 vec_{i1}, vec_{i2} 分别的权重，使得无人机在此次调整大小的过程中逼近得更加精确。

可取

$$\begin{cases} |\Delta\theta_{i1}| = \left|\theta_{i1} - \dfrac{7}{18}\pi\right| \\ |\Delta\theta_{i2}| = \left|\theta_{i2} - \dfrac{7}{18}\pi\right| \end{cases} \tag{34}$$

将两个差值归一化处理后有：

$$\begin{cases} c_1 = \dfrac{|\Delta\theta_{i1}|}{\sqrt{\Delta\theta_{i1}^2 + \Delta\theta_{i2}^2}} \\ c_2 = \dfrac{|\Delta\theta_{i1}|}{\sqrt{\Delta\theta_{i1}^2 + \Delta\theta_{i2}^2}} \end{cases} \tag{35}$$

由此可确定一次调整中的最终方向为：

$$\vec{Vec} = c_1\vec{vec}_{i1} + c_2\vec{vec}_{i2} = (c_1\cos\theta_{veci1} + c_2\cos\theta_{veci2}) \tag{36}$$

在一次迭代中，根据以上的推导可以确定无人机调整的极径和极角的变化量如下：

$$\begin{cases} \Delta r^{(k)} = \delta\left[(C_1\cos\theta_{veci1} + C_2\cos\theta_{veci2})^2 + (C_1\sin\theta_{veci1} + C_2\sin\theta_{veci2})^2\right]^{\frac{1}{2}} \\ \Delta\theta^{(k)} = \arctan\left(\dfrac{C_1\sin\theta_{veci1} + C_2\sin\theta_{veci2}}{C_1\cos\theta_{veci1} + C_2\cos\theta_{veci2}}\right) \end{cases} \tag{37}$$

其中 δ 为人为设定的调整步长，k 为迭代次数。根据以上模型，可从 $FY01$ 开始，逆时针依次根据彼此与相邻无人机的夹角 θ_{i1}, θ_{i2} 进行一次调整，当一圈无人机都完成一次调整后完成一次循环，由此所有无人机每次循环都将向最终分布的圆周上靠近，最终所有无人机位置会收敛于某一半径的圆周上，从而完成调整。

6.2 问题一（3）模型的求解

针对题目给出的数据，利用理想九边形的迭代模型对无人机位置进行调整，参数设置为 $k = 1000$，$\delta = 0.01$。

表 6　无人机极坐标数据

无人机编号	符号	半径 r (m)	角度 θ (°)
01	FY01	104.40	358.56
02	FY02	104.39	40.16
03	FY03	104.40	81.81
04	FY04	104.40	117.80
05	FY05	104.40	158.98
06	FY06	104.39	199.69
07	FY07	104.39	240.29
08	FY08	104.40	280.98
09	FY09	104.40	322.09

七、 问题二模型的建立与求解

针对问题二，首先构建与问题一 (3) 相似的理想圆迭代模型，其中变动的是从原来的正九边形变为正六边形，再假定 $FY09$ 的位置精准，作为圆心，即可使周围六个无人机调整至精确位置。在此基础上，为了确定其他位置的点依据理想圆迭代模型选取其中的菱形部分，建立三无人机的迭代调整模型，从而逐步实现其他无人机位置的调整。

7.1 问题二模型构建

7.1.1 基于理想六边形的迭代定位模型

不妨假设 $FY05$ 的位置已知,则理想情况下,以 $FY05$ 为圆心,$FY05, FY06, FY08, FY09, FY10, F$
六架无人机均匀分布一定半径的圆周上，且所有无人机位置不发生偏差，此时构成一个正六边形，则无人机与相邻无人机和中心位置无人机构成的夹角为 $\frac{\pi}{3}$, 如下图所示：

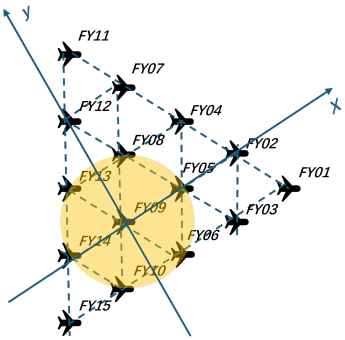


图 14　正六边形区域

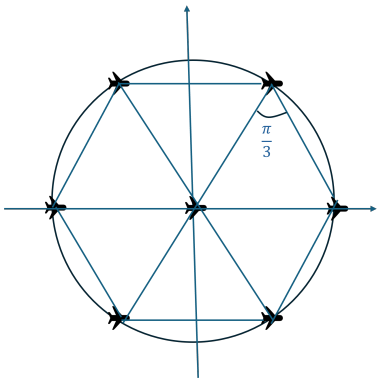


图 15　理想六边形示意图

根据理想六边形，无人机的位置偏差同样可能导致位置在九边形之外也可能在其内部，因此依据问题一 (3) 相同的方法可求得量 vec_{i1}, vec_{i2} 的具体方位，且当无人机位于 x 轴上方以及下方时，$FY0i$ 的极角 θ_i 的取值同上。

因此只需取

$$\begin{cases} |\Delta\theta_{i1}| = \left|\theta_{i1} - \dfrac{\pi}{3}\right| \\ |\Delta\theta_{i2}| = \left|\theta_{i2} - \dfrac{\pi}{3}\right| \end{cases} \tag{38}$$

将两个差值归一化处理后有：

$$\begin{cases} c_1 = \dfrac{|\Delta\theta_{i1}|}{\sqrt{\Delta\theta_{i1}^2 + \Delta\theta_{i2}^2}} \\ c_2 = \dfrac{|\Delta\theta_{i1}|}{\sqrt{\Delta\theta_{i1}^2 + \Delta\theta_{i2}^2}} \end{cases} \tag{39}$$

由此可确定一次调整中的最终方向为：

$$\vec{Vec} = c_1\vec{vec}_{i1} + c_2\vec{vec}_{i2} = (c_1\cos\theta_{veci1} + c_2\cos\theta_{veci2}) \tag{40}$$

最后可确定在一次迭代中无人机调整的极径和极角的变化量如下：

$$\begin{cases} \Delta r^{(k)} = \delta\left[(C_1\cos\theta_{veci1} + C_2\cos\theta_{veci2})^2 + (C_1\sin\theta_{veci1} + C_2\sin\theta_{veci2})^2\right]^{\frac{1}{2}} \\ \Delta\theta^{(k)} = \arctan\left(\dfrac{C_1\sin\theta_{veci1} + C_2\sin\theta_{veci2}}{C_1\cos\theta_{veci1} + C_2\cos\theta_{veci2}}\right) \end{cases} \tag{41}$$

其中 δ 为人为设定的调整步长，k 为迭代次数。根据以上模型，即可通过迭代使得六边形区域内位置偏差的无人机调整至接近理想位置。

7.1.2 基于理想菱形的迭代定位模型

注意到对于理想锥形无人机编队，可将其划分成许多小菱形单元，因此在确定以上无人机位置之后, 为了调整其他无人机的位置，可选取以下无人机进行分析:

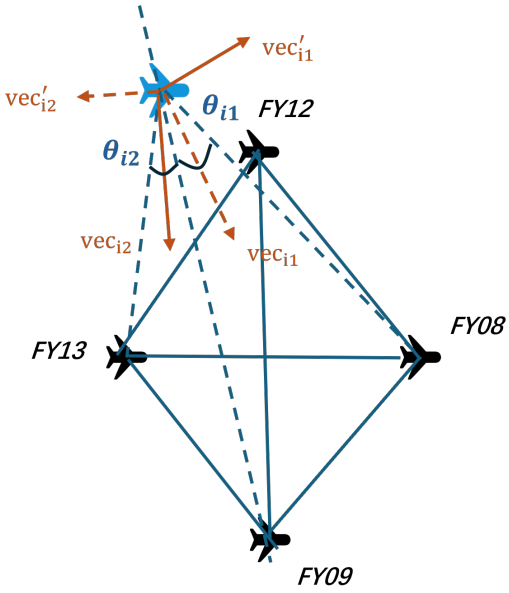


图 16　菱形调整示意图

和之前的讨论类似，当 $\theta_i 1$ 小于菱形夹角 $\frac{\pi}{5}$ 时，则向内取角平分线作为方向向量，反之则向外取角平分线作为方向向量，对于该图所示的菱形, 由于 $\theta_{i1} > \frac{\pi}{6}$,$\theta_{i2} < \frac{\pi}{6}$ 可确定:

$$\theta'_{veci1} = \frac{\theta_{i1}}{2} - \theta_i - \frac{\pi}{2} \tag{42}$$

$$\vec{vec}'_{i1} = \left(\cos\left(\theta'_{vec_{i1}}\right), \sin\left(\theta'_{vec_{i1}}\right)\right) \tag{43}$$

$$\theta_{\text{veci2}} = \theta_i - \frac{\theta_{i2}}{2} + \pi \tag{44}$$

$$\vec{vec}_{i2} = \left(\cos\left(\theta_{vec_{i2}}\right), \sin\left(\theta_{vec_{i2}}\right)\right) \tag{45}$$

因此再取

$$\begin{cases} |\Delta\theta_{i1}| = \left|\theta_{i1} - \dfrac{\pi}{6}\right| \\ |\Delta\theta_{i2}| = \left|\theta_{i2} - \dfrac{\pi}{6}\right| \end{cases} \tag{46}$$

将两个差值归一化处理后有:

$$\begin{cases} d_1 = \dfrac{|\Delta\theta_{i1}|}{\sqrt{\Delta\theta_{i1}^2 + \Delta\theta_{i2}^2}} \\ d_2 = \dfrac{|\Delta\theta_{i1}|}{\sqrt{\Delta\theta_{i1}^2 + \Delta\theta_{i2}^2}} \end{cases} \tag{47}$$

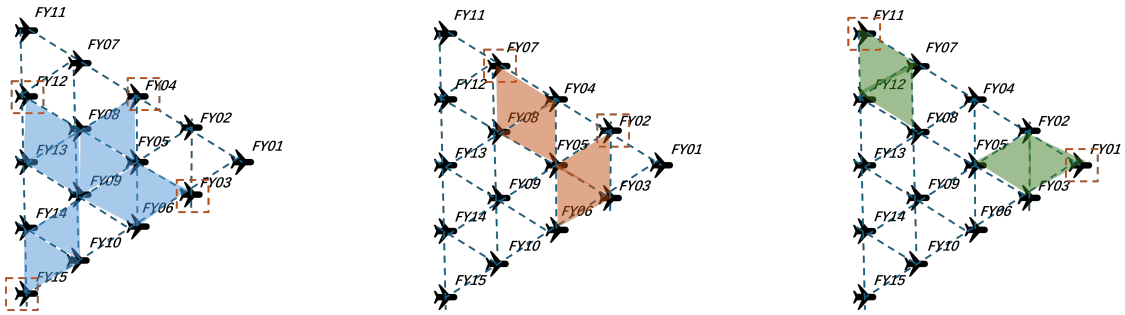
由此可确定一次调整中的最终方向为：

$$\vec{Vec} = d_1\vec{vec}_{i1} + d_2\vec{vec}_{i2} = (d_1\cos\theta_{veci1} + d_2\cos\theta_{veci2}) \tag{48}$$

最后可确定在一次迭代中 $FY12$ 调整的极径和极角的变化量如下：

$$\begin{cases} \Delta r^{(k)} = \delta\left[(d_1\cos\theta_{veci1} + d_2\cos\theta_{veci2})^2 + (d_1\sin\theta_{veci1} + d_2\sin\theta_{veci2})^2\right]^{\frac{1}{2}} \\ \Delta\theta^{(k)} = \arctan\left(\dfrac{d_1\sin\theta_{veci1} + d_2\sin\theta_{veci2}}{d_1\cos\theta_{veci1} + d_2\cos\theta_{veci2}}\right) \end{cases} \tag{49}$$

所以基于 $FY13, FY09, FY08$ 的位置，通过令 $FY12$ 反复根据所测量的方位角进行反复迭代，最终可使得其接近理想位置。根据以上思路，对剩余位置偏差的无人机利用相同的模型即可逐步调整，具体示意图如下：



(a) 过程一　　(b) 过程二　　(c) 过程三

综上，根据理想菱形调整模型，在第一轮调整中，可实现 $FY03, FY12, FY13$□$FY15$ 接近理性位置，在第二轮调整中，可实现 $FY02, FY07$ 接近理性位置，在第三轮调整中，可实现 $FY01, FY011$ 接近理性位置，最终实现整个锥形无人机编队的调整。

八、 模型的评价与推广

8.1 模型的优点

1. 模型能够处理不同的无人机阵型和不同的初始条件，适应性强，能够广泛应用于不同场景中的无人机定位与队形控制，具有较强的灵活性。

2. 基于几何学和物理学的原理（如圆周角定理、三角形的正弦定理等），模型有着严格的数学推导和理论依据，使得定位过程和结果可信且稳定，数学模型严谨。

3. 通过引入迭代调整策略，能够有效避免无人机在调整过程中的过度振荡或跳变，实现平滑的收敛过程，提升队形控制的稳定性。

8.2 模型的不足

模型对初始位置较为敏感，尤其是在初始位置偏差较大的情况下，可能导致收敛速度较慢或无法达到理想的收敛状态。

8.3 模型推广

该模型不仅可以应用于无人机编队控制，还可以扩展至其他类型的集群控制问题，如机器人集群、自动驾驶车辆的队形控制等。在复杂环境中，利用该模型可以有效提高集群的协调性和稳定性具有多项话的应用场景。

参考文献

[1] 潘兴翱, 曾杨智. 无人集群遂行城市突击的能力需求分析 [J]. 飞航导弹,2020,(11):63-67+76.DOI:10.16338/j.issn.1009-1319.20200058.

[2] 王本才, 王国宏, 何友. 多站纯方位无源定位算法研究进展 [J]. 电光与控制,2012,19(05):56-62.

[3] 季耀君. 纯方位目标定位算法的优化设计研究 [J]. 荆楚理工学院学报,2018,33(04):30-36+47.DOI:10.14151/j.cnki.jclgxyxb.2018.04.005.

[4] 王硕, 李洋, 赵蕴龙, 等. 无人机航迹规划算法综述 [J/OL]. 哈尔滨工程大学学报,1-14[2025-07-01].http://kns.cnki.net/kcms/detail/23.1390.U.20250616.1544.003.html.

附录 A　问题一第一小问

```python
import numpy as np

def locate_uav_correct(R, t, i, j, alpha1_deg, alpha2_deg):
    if i == j:
        raise ValueError("信号源编号 i 和 j 不能相同")
    if not (1 <= i <= 9 and 1 <= j <= 9):
        raise ValueError("编号必须在 1 到 9 之间")

    alpha1 = np.radians(alpha1_deg)
    alpha2 = np.radians(alpha2_deg)
    theta_i = np.radians(40 * i - 40)
    theta_j = np.radians(40 * j - 40)

    # 定义单位圆方向角
    theta_t = np.radians(40 * t - 40)

    # 顺时针方向的夹角（单位弧度）
    angle_cw1 = (theta_t - theta_i) % (2 * np.pi)
    angle_cw2 = (theta_t - theta_j) % (2 * np.pi)

    # 替代原来的 if 判断
    if angle_cw1 < np.pi:
       theta1 = np.pi/2 + theta_i - alpha1
    else:
       theta1 = -np.pi/2 + theta_i + alpha1

    if angle_cw2 < np.pi:
        theta2 = np.pi/2 + theta_j - alpha2
    else:
        theta2 = -np.pi/2 + theta_j + alpha2

    r1 = R / (2 * np.sin(alpha1))
    r2 = R / (2 * np.sin(alpha2))

```

```
    numerator = np.cos(theta2) * np.sin(alpha1) - \
        np.cos(theta1) * np.sin(alpha2)
    denominator = np.sin(theta2) * np.sin(alpha1) - \
        np.sin(theta1) * np.sin(alpha2)
    theta0_base = np.arctan2(-numerator, denominator) % (2 * np.pi)

    #  多解消除
    def total_alpha(theta_test):
        # 极角转为坐标点
        B = np.array([np.cos(theta_test), np.sin(theta_test)])  # FYOt
        A1 = np.array([np.cos(theta_i), np.sin(theta_i)])       # FYOi
        A2 = np.array([np.cos(theta_j), np.sin(theta_j)])       # FYOj
        C = np.array([0.0, 0.0])                                 # FYOO

    #  A1-B-C =
        BA1 = A1 - B
        BC = C - B
        cos1 = np.dot(BA1, BC) / (np.linalg.norm(BA1) * np.linalg.norm(BC))
        a1 = np.arccos(np.clip(cos1, -1, 1))

    #  A2-B-C =
        BA2 = A2 - B
        cos2 = np.dot(BA2, BC) / (np.linalg.norm(BA2) * np.linalg.norm(BC))
        a2 = np.arccos(np.clip(cos2, -1, 1))

        return a1 + a2

    expected_sum = alpha1 + alpha2
    candidates = [theta0_base, (theta0_base + np.pi) % (2 * np.pi)]
    theta0 = min(candidates, key=lambda th: abs(
        total_alpha(th) - expected_sum))

    cos_val = np.cos(theta0 - theta1)
    r0 = 2 * r1 * cos_val

    theta0_deg = np.degrees(theta0)
    return r0, theta0_deg


# === 主程序入口 ===
if __name__ == "__main__":
    R = 100
    t = 2
    i = 1
    j = 3
    alpha1 = 70
    alpha2 = 70

    r0, theta0 = locate_uav_correct(R, t, i, j, alpha1, alpha2)
    print(f"  定位结果: r  = {r0:.4f} m,    = {theta0:.4f}°")
```

附录 B　问题一第二小问

```python

import numpy as np

def compute_angle_range_FYOO_T_I(T_idx, I_idx, radius=15, num_samples=360):
    if T_idx == I_idx:
        raise ValueError("T 和 I 不能相同")
    R=100

    # 原始 FYOT 角度位置（单位圆上的方向）
    theta_T = np.radians(40 * T_idx - 40)
    center_T = np.array([R*np.cos(theta_T), R*np.sin(theta_T)])  # 圆心坐标

    # FYOI 位置
    theta_I = np.radians(40 * I_idx - 40)
    point_I = np.array([R*np.cos(theta_I), R*np.sin(theta_I)])  # FYOI
    point_0 = np.array([0.0, 0.0])  # FYOO

    # 采样点
    angles = np.linspace(0, 2 * np.pi, num_samples, endpoint=False)
    angle_list = []

    for phi in angles:
        # 当前采样点 T'
        T_sample = center_T + radius * np.array([np.cos(phi), np.sin(phi)])

        #  FYOO - T_sample - FYOI
        vec1 = point_0 - T_sample
        vec2 = point_I - T_sample
        dot_product = np.dot(vec1, vec2)
        norm_product = np.linalg.norm(vec1) * np.linalg.norm(vec2)
        cos_angle = np.clip(dot_product / norm_product, -1.0, 1.0)
        angle_rad = np.arccos(cos_angle)
        angle_deg = np.degrees(angle_rad)
        angle_list.append(angle_deg)

    return min(angle_list), max(angle_list)


def compute_angle_range_FYOJ_T_1(J_idx, T_idx, radius=15, num_samples=360):
    if T_idx == J_idx:
        raise ValueError("T 和 J 不能相同")
    R=100

    # 原始 FYOT 角度位置（单位圆上的方向）
    theta_T = np.radians(40 * T_idx - 40)
    center_T = np.array([R*np.cos(theta_T), R*np.sin(theta_T)])  # 圆心坐标

    # FYOJ 位置
    theta_J = np.radians(40 * J_idx - 40)
    point_J = np.array([R*np.cos(theta_J), R*np.sin(theta_J)])  # FYOJ
    point_0 = np.array([0.0, 0.0])  # FYOO

    # 采样点
    angles = np.linspace(0, 2 * np.pi, num_samples, endpoint=False)
    angle_list = []
```

```
    for phi in angles:
        # 当前采样点 T'
        T_sample = center_T + radius * np.array([np.cos(phi), np.sin(phi)])

        #  FY00 - T_sample - FY0J
        vec1 = point_0 - T_sample
        vec2 = point_J - T_sample
        dot_product = np.dot(vec1, vec2)
        norm_product = np.linalg.norm(vec1) * np.linalg.norm(vec2)
        cos_angle = np.clip(dot_product / norm_product, -1.0, 1.0)
        angle_rad = np.arccos(cos_angle)
        angle_deg = np.degrees(angle_rad)
        angle_list.append(angle_deg)

    return min(angle_list), max(angle_list)


if __name__ == "__main__":
    print("角 FY00-FY0T-FY0I (单位: 度) : ")
    for T in range(2, 10):         # T 从 2 到 9
        for I in range(1, 10):     # I 从 1 到 9
            if I == T:
                continue
            angle_min, angle_max = compute_angle_range_FY00_T_I(T, I)
            print(f"T = {T}, I = {I} → FY00-FY0{T}-FY0{I} = [{angle_min:.4f}°, {angle_max:.4f}°]")

    print("角 FY0J-FY0T-FY01 (单位: 度) : ")
    for J in range(2, 10):         # J 从 2 到 9
        for T in range(2, 10):     # T 从 2 到 9
            if J == T:
                continue
            angle_min, angle_max = compute_angle_range_FY0J_T_1(J, T)
            print(f"J = {J}, T = {T} → FY0{J}-FY0{T}-FY01 = [{angle_min:.4f}°, {angle_max:.4f}°]")

import numpy as np

def compute_angle_FY00_T_I(T, I):
    if T == I:
        raise ValueError("T 和 I 不能相同")

    # 每个点的角度 (以 40° 间隔, 转换为弧度)
    theta_T = np.radians(40 * T - 40)
    theta_I = np.radians(40 * I - 40)

    # 极坐标单位向量表示
    P0 = np.array([0.0, 0.0])  # 原点 FY00
    PT = np.array([np.cos(theta_T), np.sin(theta_T)])  # FY0T
    PI = np.array([np.cos(theta_I), np.sin(theta_I)])  # FY0I

    #  P0 - PT - PI
    vec1 = P0 - PT
    vec2 = PI - PT

    dot_product = np.dot(vec1, vec2)
    norm_product = np.linalg.norm(vec1) * np.linalg.norm(vec2)
```

```
    cos_angle = np.clip(dot_product / norm_product, -1.0, 1.0)
    angle_rad = np.arccos(cos_angle)
    angle_deg = np.degrees(angle_rad)

    return angle_deg

if __name__ == "__main__":
    print("角 FY00-FY0T-FY0I (单位: 度) : ")
    for T in range(2, 10):         # T 从 2 到 9
        for I in range(1, 10):     # I 从 1 到 9
            if I == T:
                continue
            angle = compute_angle_FY00_T_I(T, I)
            print(f"T = {T}, I = {I} → FY00-FY0{T}-FY0{I} = {angle:.4f}°")

    print("角 FY0J-FY0T-FY01 (单位: 度) : ")
    for J in range(2, 10):         # J 从 2 到 9
        for T in range(2, 10):     # T 从 2 到 9
            if J == T:
                continue
            angle = compute_angle_FY00_T_I(J, T)
            print(f"J = {J}, T = {T} → FY0{J}-FY0{T}-FY01 = {angle:.4f}°")

import numpy as np


def compute_angle_range_FY00_T_I(T_idx, I_idx, radius=15, num_samples=360):
    if T_idx == I_idx:
        raise ValueError("T 和 I 不能相同")
    R = 100

    # 原始 FY0T 角度位置 (单位圆上的方向)
    theta_T = np.radians(40 * T_idx - 40)
    center_T = np.array([R*np.cos(theta_T), R*np.sin(theta_T)])  # 圆心坐标

    # FY0I 位置
    theta_I = np.radians(40 * I_idx - 40)
    point_I = np.array([R*np.cos(theta_I), R*np.sin(theta_I)])  # FY0I
    point_0 = np.array([0.0, 0.0])  # FY00

    # 采样点
    angles = np.linspace(0, 2 * np.pi, num_samples, endpoint=False)
    angle_list = []

    for phi in angles:
        # 当前采样点 T'
        T_sample = center_T + radius * np.array([np.cos(phi), np.sin(phi)])

        #  FY00 - T_sample - FY0I
        vec1 = point_0 - T_sample
        vec2 = point_I - T_sample
        dot_product = np.dot(vec1, vec2)
        norm_product = np.linalg.norm(vec1) * np.linalg.norm(vec2)
        cos_angle = np.clip(dot_product / norm_product, -1.0, 1.0)
        angle_rad = np.arccos(cos_angle)
        angle_deg = np.degrees(angle_rad)
```

```
        angle_list.append(angle_deg)

    return min(angle_list), max(angle_list)


def compute_angle_range_FY0J_T_1(J_idx, T_idx, radius=15, num_samples=360):
    if T_idx == J_idx:
        raise ValueError("T 和 J 不能相同")
    R = 100

    # 原始 FY0T 角度位置（单位圆上的方向）
    theta_T = np.radians(40 * T_idx - 40)
    center_T = np.array([R*np.cos(theta_T), R*np.sin(theta_T)])  # 圆心坐标

    # FY0J 位置
    theta_J = np.radians(40 * J_idx - 40)
    point_J = np.array([R*np.cos(theta_J), R*np.sin(theta_J)])  # FY0J
    point_0 = np.array([0.0, 0.0])  # FY00

    # 采样点
    angles = np.linspace(0, 2 * np.pi, num_samples, endpoint=False)
    angle_list = []

    for phi in angles:
        # 当前采样点 T'
        T_sample = center_T + radius * np.array([np.cos(phi), np.sin(phi)])

        #  FY00 - T_sample - FY0J
        vec1 = point_0 - T_sample
        vec2 = point_J - T_sample
        dot_product = np.dot(vec1, vec2)
        norm_product = np.linalg.norm(vec1) * np.linalg.norm(vec2)
        cos_angle = np.clip(dot_product / norm_product, -1.0, 1.0)
        angle_rad = np.arccos(cos_angle)
        angle_deg = np.degrees(angle_rad)
        angle_list.append(angle_deg)

    return min(angle_list), max(angle_list)


def locate_uav_correct(R, t, i, j, alpha1_deg, alpha2_deg):
    if i == j:
        raise ValueError("信号源编号 i 和 j 不能相同")
    if not (1 <= i <= 9 and 1 <= j <= 9):
        raise ValueError("编号必须在 1 到 9 之间")

    alpha1 = np.radians(alpha1_deg)
    alpha2 = np.radians(alpha2_deg)
    theta_i = np.radians(40 * i - 40)
    theta_j = np.radians(40 * j - 40)

    # 定义单位圆方向角
    theta_t = np.radians(40 * t - 40)

    # 顺时针方向的夹角（单位弧度）
    angle_cw1 = (theta_t - theta_i) % (2 * np.pi)
```

```
    angle_cw2 = (theta_t - theta_j) % (2 * np.pi)

    # 替代原来的 if 判断
    if angle_cw1 < np.pi:
       theta1 = np.pi/2 + theta_i - alpha1
    else:
       theta1 = -np.pi/2 + theta_i + alpha1

    if angle_cw2 < np.pi:
        theta2 = np.pi/2 + theta_j - alpha2
    else:
        theta2 = -np.pi/2 + theta_j + alpha2

    r1 = R / (2 * np.sin(alpha1))
    r2 = R / (2 * np.sin(alpha2))

    numerator = np.cos(theta2) * np.sin(alpha1) - \
        np.cos(theta1) * np.sin(alpha2)
    denominator = np.sin(theta2) * np.sin(alpha1) - \
        np.sin(theta1) * np.sin(alpha2)
    theta0_base = np.arctan2(-numerator, denominator) % (2 * np.pi)

    #  多解消除
    def total_alpha(theta_test):
        # 极角转为坐标点
        B = np.array([np.cos(theta_test), np.sin(theta_test)])  # FYOt
        A1 = np.array([np.cos(theta_i), np.sin(theta_i)])       # FYOi
        A2 = np.array([np.cos(theta_j), np.sin(theta_j)])       # FYOj
        C = np.array([0.0, 0.0])                                 # FYOO

    #  A1-B-C =
        BA1 = A1 - B
        BC = C - B
        cos1 = np.dot(BA1, BC) / (np.linalg.norm(BA1) * np.linalg.norm(BC))
        a1 = np.arccos(np.clip(cos1, -1, 1))

    #  A2-B-C =
        BA2 = A2 - B
        cos2 = np.dot(BA2, BC) / (np.linalg.norm(BA2) * np.linalg.norm(BC))
        a2 = np.arccos(np.clip(cos2, -1, 1))

        return a1 + a2

    expected_sum = alpha1 + alpha2
    candidates = [theta0_base, (theta0_base + np.pi) % (2 * np.pi)]
    theta0 = min(candidates, key=lambda th: abs(
        total_alpha(th) - expected_sum))

    cos_val = np.cos(theta0 - theta1)
    r0 = 2 * r1 * cos_val

    theta0_deg = np.degrees(theta0)
    return r0, theta0_deg


if __name__ == "__main__":
```

```
    alpha1 = 70
    alpha2 = 70

    T = 2

    for I in range(2, 10):     # I 从 2 到 9
        if I == T:
            continue
        angle_min1, angle_max1 = compute_angle_range_FY00_T_I(T, I)
        angle_min2, angle_max2 = compute_angle_range_FY0J_T_1(I, T)

        if angle_min1 <= alpha1 <= angle_max1 and angle_min2 <= alpha2 <= angle_max2:
            print(f"  符合观测角度的编号: I = J = {I} 固定")

            R = 100

            r0, theta0 = locate_uav_correct(R, T, I, 1, alpha1, alpha2)
            print(f"  定位结果: r  = {r0:.4f} m,    = {theta0:.4f}°")
```

附录 C　问题一第三小问

```

import numpy as np

# === 参数设置 === #
theta_ideal = np.pi * 7 / 18    # 理想角 70°
delta = 0.01                     # 每轮迭代步长
max_iter = 1000                  # 最大迭代轮数

def polar_to_cartesian(r, theta_deg):
    theta_rad = np.radians(theta_deg)
    return r * np.array([np.cos(theta_rad), np.sin(theta_rad)])

def angle_between_vectors(v1, v2):
    cos_theta = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
    return np.arccos(np.clip(cos_theta, -1, 1))

def normalize_angle(theta):
    return theta % (2 * np.pi)

def get_unit_vector(angle_rad):
    return np.array([np.cos(angle_rad), np.sin(angle_rad)])


#迭代修正函数
def iterate_position_correction(initial_positions, center=np.array([0.0, 0.0]), target_r=None):
    positions = initial_positions.copy()
    n = len(positions)
    if target_r is None:
        # 默认目标半径为初始极径均值
        target_r = np.mean([np.linalg.norm(p - center) for p in positions])

    for iteration in range(max_iter):
```

```
        print(f"\n 第 {iteration + 1} 轮迭代")
        new_positions = positions.copy()

        for i in range(n):  # 所有无人机都参与
            Pi = positions[i]
            Pi_prev = positions[(i - 1) % n]
            Pi_next = positions[(i + 1) % n]

            # 向量定义
            vec_prev = Pi_prev - Pi
            vec_center = center - Pi
            vec_next = Pi_next - Pi

            # 两个夹角
            theta1 = angle_between_vectors(vec_prev, vec_center)
            theta2 = angle_between_vectors(vec_next, vec_center)

            # 偏离程度（角度偏差归一化）
            d1 = abs(theta1 - theta_ideal)
            d2 = abs(theta2 - theta_ideal)
            norm = np.sqrt(d1**2 + d2**2) + 1e-6
            c1, c2 = d1 / norm, d2 / norm

            # 两个角平分线方向（修正）
            bisector1 = (vec_prev / np.linalg.norm(vec_prev) + vec_center / np.linalg.norm(vec_center))
            bisector1 /= np.linalg.norm(bisector1)
            bisector2 = (vec_next / np.linalg.norm(vec_next) + vec_center / np.linalg.norm(vec_center))
            bisector2 /= np.linalg.norm(bisector2)
            direction = c1 * bisector1 + c2 * bisector2
            direction /= np.linalg.norm(direction)

            # 径向修正分量
            r_now = np.linalg.norm(Pi - center)
            radial_dir = (Pi - center) / (r_now + 1e-8)
            radial_correction = (target_r - r_now) * 0.2 * radial_dir  # 0.2为径向收敛速率

            move_vec = delta * direction + radial_correction
            new_positions[i] += move_vec

            print(f"FY0{i+1}:  1={np.degrees(theta1):.2f}°,  2={np.degrees(theta2):.2f}°, dx={move_vec[0]:.3f}, dy
    ={move_vec[1]:.3f}, r={np.linalg.norm(new_positions[i]-center):.2f}")

        positions = new_positions.copy()

    return positions


# === 主程序入口 === #

if __name__ == "__main__":
    # 初始极坐标位置（单位：米 / 度）
    init_polar = {
        1: (100, 0),
        2: (98, 40.10),
        3: (112, 80.21),
        4: (105, 119.75),
```

```
        5: (98, 159.86),
        6: (112, 199.96),
        7: (105, 240.07),
        8: (98, 280.17),
        9: (112, 320.28)
    }

    # 转换为直角坐标
    initial_positions = [polar_to_cartesian(r, theta) for r, theta in init_polar.values()]

    # 执行迭代修正
    corrected_positions = iterate_position_correction(initial_positions)

    # 输出最终位置（极坐标形式）
    print("\n 最终修正位置坐标：")
    for i, pos in enumerate(corrected_positions):
        r = np.linalg.norm(pos)
        theta = np.degrees(np.arctan2(pos[1], pos[0])) % 360
        print(f"FY0{i+1}: x={pos[0]:.2f}, y={pos[1]:.2f}, r={r:.2f}, ={theta:.2f}°")
```