基于动态规划迭代思想的无人机飞行问题研究

摘要

无人机集群在遂行编队飞行时，存在外界干扰，定位紊乱的问题，如何在保持电磁静默的情况下以**最少**的电磁信号实现最精准的定位就成了重中之重，同时为了保持编队队形，采用**纯方位无源定位**的方法调整无人机的位置。

问题一第一小问即是在发射信号的无人机位置精确的情况下，对接受信号的无人机建立定位模型。而**问题一第二小问**则是求需要几架发射信号的无人机才能实现无人机的有效定位。**问题一第三小问**是利用数据和方向信息，给出具体的调整方案。**问题二是对问题一**的延伸，采用其他编队队形，仍考虑纯方位无人定位的情形下，设计无人机的调整方案。

针对问题一第一小问，因为发射信号的 3 架无人机位置无偏差且编号已知，可以得到这 3 架无人机的方向信息。本文利用**三角定位法**与**余弦定理**，列出 3 架无人机所构建的附加圆的方程组，联立求解得出结果，从而得到被动接收信号的无人机的定位模型。

针对问题一第二小问，因为无法知道除 FY00 与 FY01 外的其他无人机的编号，不能简单地使用三角定位来求解，本文采用多个附加圆交叉定位的方法求解。

针对问题一第三小问，因为发送信号的无人机位置有偏差，为在发送信号最少的情况下调整无人机位置，本文采取**无限逼近最优解的贪心迭代算法**，通过运用**贪心思想**计算出偏移量最小的点，从而得到一个最接近标准圆的圆，也就是当前最优解，以此为标准去移动剩余 7 架圆周飞机，使其**无限靠近**最优解圆周，再比较 FY02-FY09 中的偏移量最小的点，得到一个更加接近标准圆的圆，去调整其他飞机。

如此往复使得所有圆周飞机无限地接近于标准圆周上，再计算圆周上角度偏差最小的点近似认为标准点与 FY01 组成发射端，去调整 FY0H 的偏移角(H 为发射端以外的圆周飞机)，保持发射端所形成的信号角**不变**，依据圆周角不变来确保被调整飞机不脱离圆周。再**迭代求最优点**，使得所有点无限接近于标准点。最终解得本题。

针对问题二，因为需调整为锥形队形，而相邻的两架无人机间距相等，故理想状态下应该是等边三角形，本文通过**优化偏差值算法**尽可能的调整无人机的偏差，先将三角形框架的顶点尽可能的标准化，再以此为标准，将内一层的三角形框架顶点标准化，以此类推直到最内层的三角形，再从最内层的三角形顶点向外层顶点进行优化，同时调整边上的点。所得结果准确度较高，能**二次优化**数据，且可用于规模更大的锥形无人机阵列。

关键词：三角测量定位法，有效定位分析，贪心，迭代算法，动态规划，二次优化

一、 问题重述

1.1 问题背景

由于无人机集群在遂行编队飞行时, 应尽可能的避免外界干扰,采用纯方位无源定位的方法调整无人机的位置, 即由编队中某几架无人机发射信号, 其余无人机被动接收信号, 从中提取出方向信息进行定位, 来调整无人机的位置。

1.2 问题重述

本文将要解决以下几个问题：

问题一：10 架无人机组成圆形编队，1 架位于圆心的无人机与 9 架均匀排布于圆周上的无人机组成圆形编队，无人机基于自身感知高度信息数据，均在同一高度平面上飞行。建立数学模型, 解决以下问题：

1、 位于圆心的 1 架无人机与位于圆周上的 2 架无人机发射信号，其余无人机接收信号，但位置稍有偏差。拟定发射信号的 3 架无人机编号已知并且无偏差，建立被动接收信号的无人机的定位模型。

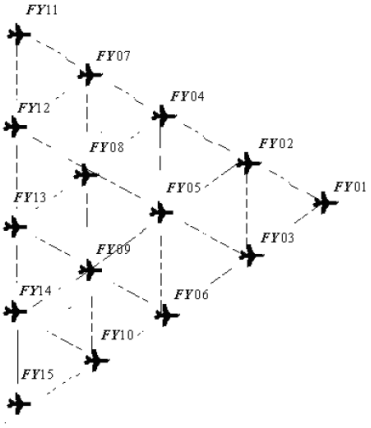
2、1 架位置稍有偏差的无人机接收到编号为 FY00 和 FY01 的无人机发射出的信号，同时还接受到圆周上若干架无人机发射出的信号，若发射信号的无人机的位置毫无偏差，问除了 FY00 和 FY01 2 架无人机外还需几架无人机发射出的信号才能实现无人机的准确定位。

3、 1 架位于圆心的无人机与 9 架均匀排布在直径为 200m 的圆周上的无人机组成编队。当初始时刻无人机的位置稍有偏差时，给出合理的位置调整策略，使其多次调整，每次选择圆心的无人机与圆周上不超过 3 架无人机遂行发射位置信号，剩余的无人机根据发射出的位置信号，调整到理想位置(每次调整的时间忽略不计)，实现 9 架无人机最终均匀排布在某个圆周上。利用表 1 所提供的数据，只依照接受到的位置信号来调整无人机的位置信号，要求给出具体的调整策略。

表 1　　无人机的初始位置表

无人机编号	极坐标 (m,∘)
0	(0, 0)
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)

问题二：实际飞行的过程，无人机排布也可以是其他的队形，例如锥形编队队形（见图 3，直线上相邻两架无人机的间距相等，如 50m）。依然考虑纯方位无源定位的情形，设计无人机位置调整方案。



二、 问题分析

2.1 问题一的分析

问题一主要围绕十架无人机形成圆形编队且保持同一高度飞行时,通过若干无人机发射的信号实现纯方位无源定位的问题。只需要考虑平面上的圆周问题.根据问题一的假设,剩下的九架飞机尽量均匀分布在圆周上。对于位置准确的无人机,其位置仅与无人机的编号有关；而对于位置略有偏差的无人机,他们距离圆心无人机的距离以及角度都是未知的。

现在我们需要对问题一的三个小问,进行分析：

2.1.1 问题一第一小问

第一小问中当位于圆心无人机（FY00）和编队中已知编号且位置无偏差的两架无人机发射信号, 建立被动接收信号无人机的定位模型.对该问题,我们三角测量定位法进行求解：

以三个圆的交点确定接收无人机的位置，列出三个圆的方程组，联立方程组，转化为求解一个最小二乘问题，求得无人机的位置。

2.1.2 问题一第二小问。

第二小问问题中我们无法得知除 FY00 和 FY01 外其余发射信号的无人机的编号,无法直接利用三角测量定位法进行求解.因此使用交叉定位方法进行求解, 我们逐一增加位置略有偏移的发射无人机的数目,通过增加的无人机列出方程，消去未知数，直到可以确定无人机的位置。

2.1.3 问题三第三小问

第三小问需要对初始时刻位置存在偏差的无人机的位置进行调整.由前两问已知实现无人机的有效定位, 至少需要圆周上两架无人机发送信号.由于发送信号的无人机位置存在偏差, 因此我们可以每次求出最优点以更新其他点的值。

2.2 问题二的分析

问题二可以采用动态规划的思想分析。与问题一不同的是, 问题二的无人机一定在同一高度飞行,我们可以通过每次寻找最优点来调整其它的点, 再通过新得到的最优点再次调整其他点，从而求得使无人机在调整过程中偏差尽可能小的方案。

三、 模型假设

假设1：题目中的无人机略有偏差时,不存在角度偏离超过$\frac{2\pi}{9}$范围的无人机。

假设2：相较于无人机的距离，无人机大小可以忽略不计

假设3：假设无人机在飞行时，偏差位置不会改变。

假设4：假设在对无人机偏差位置进行调整时，不需要考虑调整时间。

假设5：假设每架无人机标号都是固定的, 且明确知道哪些无人机发射信号。

四、 符号说明

符号	符号意义
$\alpha_k, k=1,2,3,...$	接收信号无人机与发射信号无人机的夹角
$(m_i, n_i), i=1,2,3,...$	外接圆圆心坐标
P	被动接收信号无人机
$r_i, i=1,2,3,...$	外接圆半径
K	无人机标号
θ	信息角

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 第一问——三角测量定位法求解具体位置[1]

假设圆周的半径为 R，以无人机 FY00 为坐标原点，连接 FY01 方向的射线为 x 轴，垂直于 x 轴竖直向上的方向为 y 轴建立平面直角坐标系，不失一般性。假设三个发射信号的无人机分别为 FY00(0，0)，FY01(R,0),FY0k(Rcosθ, Rsinθ)，其中 K≠0 和 K≠1。建立平面直角坐标系定位模型如下：

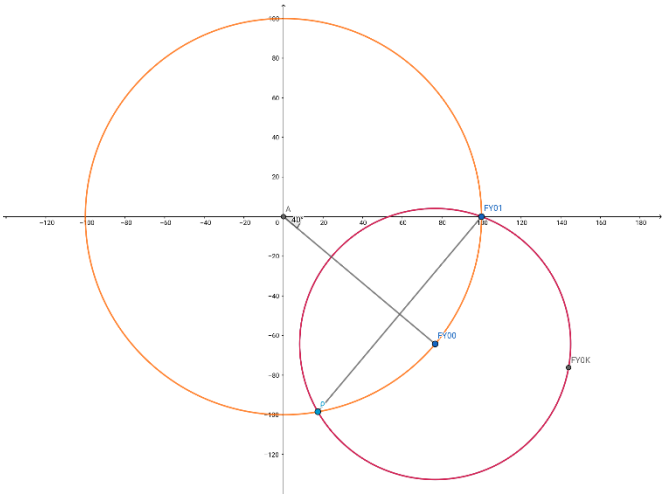


图 5.1 无人机分布三角定位模型——无误差

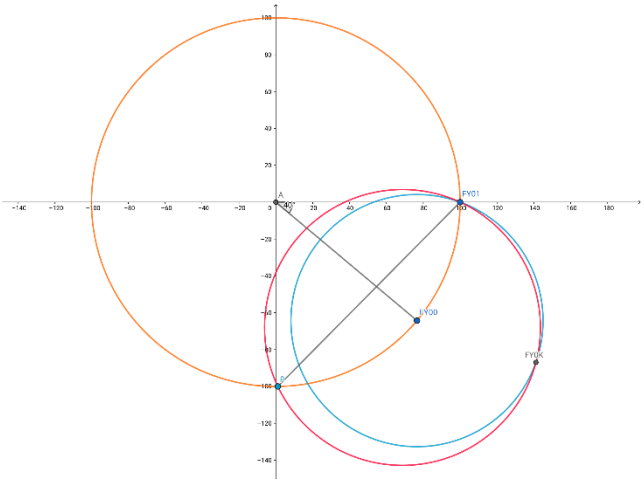


图 5.2 无人机分布三角定位模型——有误差

已知圆上两点，可以确定该圆的圆心角。按照 FY00(0,0),FY01(R,0)的位置和该圆的圆心角，确定圆心的位置，从而绘出附加圆的轨迹设附加圆的圆心 O(m_1, n_1)，半径为r_1，则附加圆的方程为

$$(m - m_1)^2 + (n - n_1)^2 = {r_1}^2 \quad (5\text{-}1)$$

FY00(0,0),FY01(R,0)两点坐标与圆心 O 所构成的三角形，以及余弦定理，推出方程组

$$\begin{cases} （m_1 - 0）^2 + n_1^2 = r_1^2 \\ (m_1 - R)^2 + n_1^2 = r_1^2 \\ R^2 = 2r_1^2(1 - cos\,2\,\alpha_1) \end{cases} \quad (5\text{-}2)$$

联立推算得到附加圆的圆心坐标(m_1,n_1)与其半径 r_1

同理，我们可以用 FY00(0，0)，FY01(R,0),FY0k(Rcosθ, Rsinθ)这三个点中两点的其他组合得到如下关系

$$\begin{cases} (m_2 - 0)^2 + n_2^2 = r_2^2 \\ (m_2 - R\cos\theta)^2 + (n_2 - R\sin\theta)^2 = r_2^2 \\ R^2 = 2r_2^2(1 - \cos 2\alpha_2) \end{cases} \quad (5\text{-}3)$$

$$\begin{cases} (\mathrm{m}_3 - R\cos\theta)^2 + (n_3 - R\sin\theta)^2 = r_3^2 \\ (m_3 - R)^2 + n_3 = r_3^2 \\ (R - R\cos\theta)^2 + (1 - R\sin\theta)^2 = 2r_3^2(1 - \cos 2\alpha_3)^2 \end{cases} \tag{5-4}$$

联立推算得到(m_2,n_2)与 r_2 和(m_3,n_3)与 r_3，方程组为：

$$\begin{cases} \mathrm{r}_1^2 = \frac{R^2}{2(1-\cos(2\alpha_1))} \\ m_1^2 = \frac{R^2}{4}, \\ n_1^2 = \frac{R^2}{2(1-\cos(2\alpha_1))} - \frac{R^2}{4}, \end{cases} \tag{5-5}$$

$$\begin{cases} \mathrm{r}_2^2 = \frac{R^2}{2(1-\cos(2\alpha_2))}, \\ n_2 = \sqrt{\mathrm{r}_2^2 + R^2\left(\frac{1}{2} - \frac{\sin(\theta)^2}{4}\right)} - \frac{R}{2}\sin(\theta), \\ m_2 = \frac{R}{2\cos(\theta)} + \tan(\theta)y_2, \end{cases} \tag{5-6}$$

$$\begin{cases} r_3^2 = \frac{R^2(1-\cos(\theta))^2+(1-\sin(\theta)R)^2}{2(1-\cos(2\alpha_3))} \\ n_3 = \frac{1-\cos(\theta)}{\sin(\theta)}x_3, \\ m_3 = \frac{R}{\left(1+\frac{(1-\cos(\theta))^2}{\sin(\theta)^2}\right)} + \sqrt{\frac{R^2}{\left(1+\frac{(1-\cos(\theta))^2}{\sin(\theta)^2}\right)} - \frac{r_3^2}{\left(1+\frac{(1-\cos(\theta))^2}{\sin(\theta)^2}\right)}} \end{cases} \tag{5-7}$$

得到三个附加圆的轨迹后，我们为求三个附加圆的交点坐标，也就是接收信号的无人机的位置，联立可得：

$$\begin{cases} (m_1 - m)^2 + (n_1 - n)^2 = r_1 \\ (m_2 - m)^2 + (n_2 - n)^2 = r_2 \\ (m_3 - m)^2 + (n_3 - n)^2 = r_3 \end{cases} \tag{5-8}$$

展开化简

$$\begin{bmatrix} r_1^2 - K_1 \\ r_2^2 - K_2 \\ r_3^2 - K_3 \end{bmatrix} = \begin{bmatrix} -2m_1 & -2n_1 & 1 & m \\ -2m_2 & -2n_2 & 1 & n \\ -2m_3 & -2n_3 & 1 & Z \end{bmatrix} \tag{5-9}$$

其中，$K_i = m_i^2 + n_i^2$，$Z = m^2 + n^2$，$i = 1,2,3$

将方程记为：

$$N = AM \tag{5-10}$$

便可将问题转化为最小二乘问题：

$$\min_{M} \|N - AM\|_2^2 \tag{5-11}$$

最终得到解析式为：$M = (A^T A)^{-1} A^T N$。如此便建立起了被动接收信号无人机的定位模型。

5.1.2 第二问——无人机有效定位分析[2]

倘若只有一架编号未知的无人机发射信号，无法满足无人机的有效定位，因此我们对于两架无人机进行考虑

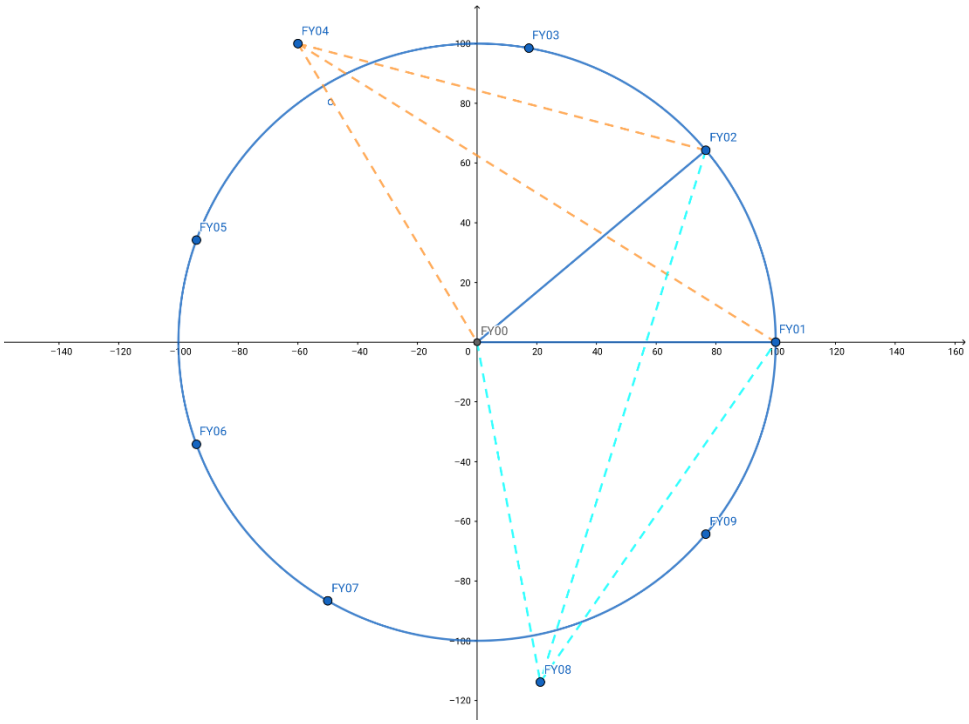


图 5.3 四架无人机定位图

由于第二小问中，圆周上发射信号的无人机除 FY00, FY01 外，编号均是未知的，因此，使用双站交叉定位计算方法。设圆周上两架发射信号的无人机为：ρ_n ρ_m ,其坐标分别为$(R\cos((m-1)\alpha), R\sin((m-1)\alpha))$以及$(R\cos((n-1)\alpha), R\sin((n-1)\alpha))$，其中$m \neq n$，$m, n \neq 0, 1$。

将二者看作已知,每 3 个点确定一个定圆，求得圆心一定点对四架发射信号的无人机逐一计算，可以得到四个附加圆，关于 m，n，以及二者坐标的表达式。

此时求出的四个动点必定为同一点，消去 m，n，从而得到该动点的具体位置，分类情况如下表：

5.1 分类情况

分类	无人机编号		
1	FY00	FY01	$FY0K_1$
2	FY00	FY01	$FY0K_2$
3	FY00	$FY0K_1$	$FY0K_2$
4	FY01	$FY0K_1$	$FY0K_2$

分类如下：

分类 1 FY00，FY01 是已知点，α_1已知，则每两个发射端可以与接收端确定一个圆的范围，所以有C_3^2个圆，又三个圆可以确定一个交点，这个交点是由m_1，n_1，r_1，m和组成的方程。

同理可得：

分类 2 可推出由m_2，n_2，r_2，n组成的关于一个交点的方程；

分类 3 可推出由m_3，n_3，r_3，m，n组成的关于一个交点的方程；

分类 4 可推出由m_4，n_4，r_4，m，n组成的关于一个交点的方程。

由以下推导所得的式子可得，m_i，n_i，r_i是关于已知量R和α_i的已知量$(i=(1，2，3，4))$所以根据解的唯一性，可以得出四个交点必定为同一点，故可消去所取得两架发射无人机的点位中所带的未知量，解出接收无人机的有效位置。

步骤如下

1、由 FY00(0,0),FY01(R,0)坐标以及二者之间的夹角，可以确定一个唯一的圆，圆心为O_1,坐标为(m_1,n_1),半径为r_1

得到方程组：

$$\begin{cases}(m_1-0)^2+n_1^2=r_1^2\\(m_1-R)^2+n_1^2=r_1^2\\R^2=2r_1^2(1-\cos 2\alpha_1)\end{cases}\tag{5-12}$$

从而求解出(m_1,n_1),与r_1。

$$\begin{cases}\mathrm{r}_1^2=\frac{R^2}{2(1-\cos(2\alpha_1))}\\m_1^2=\frac{R^2}{4},\\n_1^2=\frac{R^2}{2(1-\cos(2\alpha_1))}-\frac{R^2}{4},\end{cases}\tag{5-13}$$

同理，先前所设位置准确但编号未知的两架无人机ρ_n ρ_m，此时的 m，n 均未知。与第一个附加圆的推导相似的，我们有

$$\begin{cases}(m_2-0)^2+n_2^2=r_2^2\\(m_2-R\cos((m-1)\alpha))^2+(n_2-R\sin((n-1)\alpha))^2=r_2^2\\R^2=2r_2^2(1-\cos(2\alpha_2))\end{cases}\tag{5-14}$$

并由此解得(m_2,n_2),以及r_2。

同理

$$\begin{cases}(m_3-R\cos((m-1)\alpha))^2+(n_3-R\sin((n-1)\alpha))^2=r_3^2\\(m_3-R)^2+n_3^2=r_3^2\\(R-R\cos((m-1)\alpha))^2+(1-R\sin((n-1)\alpha))^2=2r_3^2(1-\cos(2\alpha_3))\end{cases}\tag{5-15}$$

可求得圆心坐标(m_3,n_3)以及r_3：

$$\begin{cases}r_3^2=\frac{R^2(1-\cos((m-1)\alpha))^2+(1-\sin((m-1)\alpha)R)^2}{2(1-\cos(2\alpha_3))}\\n_3=\frac{1-\cos((m-1)\alpha)}{\sin((m-1)\alpha)}m_3,\\m_3=\frac{R}{\left(1+\frac{(1-\cos((m-1)\alpha))^2}{\sin((m-1)\alpha)^2}\right)}+\sqrt{\frac{R^2}{\left(1+\frac{(1-\cos((m-1)\alpha))^2}{\sin((m-1)\alpha)^2}\right)}-\frac{r_3^2}{\left(1+\frac{(1-\cos((m-1)\alpha))^2}{\sin((m-1)\alpha)^2}\right)}}\end{cases}\tag{5-16}$$

同理可得与(m_4,n_4)以及半径r_4相关方程组：

$$\begin{cases}(m_4-R\cos(k_2\alpha))^2+(n_4-R\sin((n-1)\alpha))^2=r_4^2\\(m_4-R)^2+n_4^2=r_4^2\\(R-R\cos((n-1)\alpha))^2+(1-R\sin((n-1)\alpha))^2=2r_4^2(1-\cos(2\alpha_4))\end{cases}\tag{5-17}$$

从而可得(m_4,n_4)以及半径r_4。

2、目的是得到动点的坐标，将上述所得元素联立方程组可得：

$$\begin{cases}(m_1-m)^2+(n_1-n)^2=r_1^2\\(m_2-m)^2+(n_2-n)^2=r_2^2\\(m_3-m)^2+(n_3-n)^2=r_3^2\\(m_4-m)^2+(n_4-n)^2=r_4^2\end{cases}\tag{5-18}$$

3、因为无人机的位置仅仅稍有偏差，所以能够通过ρ_n ρ_m发射的信号判断他们之间的夹角，得到附加圆的圆心角。哪怕只是稍有偏差，也足以证明二者的独立性，通过二者的差来判断ρ_n ρ_m的关系。

将问题梳理后我们可以得知，利用动点实际为一点建立起的方程组，我们消去 m，n 得到动点的坐标，即可确定动点的有效定位，因此除 FY00 与 FY01 之外，还需要额外的两架无人机发射位置信号，方能实现无人机的有效定位。[3]

5.1.3 第三问——动态规划分析分布策略

该问题要求我们在初始时刻无人机的位置稍有偏差时给出合理的位置调整方案，要求圆周上不超过 3 架无人机，最终实现无人机均匀排布在某个圆周上。

根据表 1-无人机的初始位置表，拟定 00 号无人机的位置作为一个标准圆的圆心，其与 01 号无人机位置的连线为半径作出一个半径为 100m 的标准圆。其他点在这基础上调整，使得有偏差的点尽可能高效地接近标准圆。

飞行过程中需要尽可能保持电磁静默，最大化减少发射电磁波信号。

1、一定条件下错误的简易初步模型

由题意知，FY00、FY01 为标准点位，则可以 FY00 为圆心，FY00 与 FY01 的连线为半径，得到一个标准圆，FY0K'为一偏差点，现尝试将 FY0K'移动到标准圆周上，由 FY00,FY01 组成发射端，得信息角θ_1，因为 FY0K'坐标已知，标准圆半径已知，则θ_2可求，故而只需要将 FY0K 沿着极径移动到标准圆周上，无人机判断到达目标点的依据是θ_1=θ_2。如此可依次把偏差点移动到标准圆圆周，再调整偏差角。

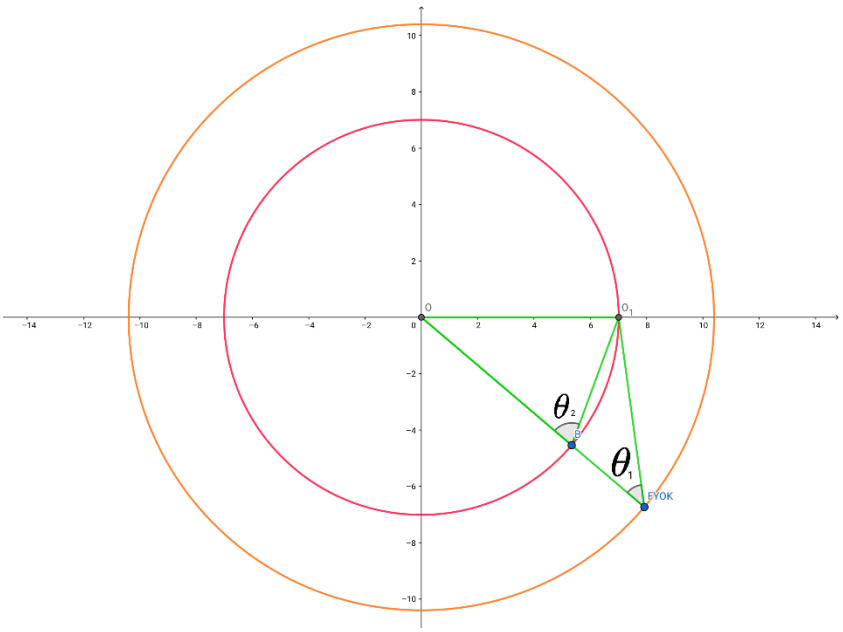


图 5.4 简易初步模型图

(1)错误条件分析

该算法看似可求，但根据题意，无人机的移动依据只能是无人机自身所得信号角，而根据无人机自身所得信号角θ_1，无人机无法获知θ_2的大小，则无法判断调整方向。故此时条件不成立。

2.无限逼近最优解的贪心的迭代算法

由题意知，FY00、FY01 为标准点位，则可以 FY00 为圆心，FY00 与 FY01 的连线为半径，得到一个标准圆。

此时采用贪心思想，从 FY02-FY09 中，计算出偏移量最小的点。从而得出一个最接近标准圆的圆，为当前的最优解。

以当前最优解的圆为标准，去移动剩余 7 架圆周飞机，使其无限靠近最优解圆周，再比较 FY02-FY09 中的偏移量最小的点，得到一个更加接近标准圆的圆，去调整其他飞机，如此往复使得所有圆周飞机无限地接近于标准圆周上。

再计算圆周上角度偏差最小的点近似认为标准点与 FY01 组成发射端，去调整 FY0H 的偏移角(H 为发射端以外的圆周飞机)，保持发射端所形成的信号角不变，依据圆周角不变来确保被调整飞机不脱离圆周。

最后迭代求最优点，使得所有点无限接近于标准点。最终解得本题。

计算模型如下:

1、x 轴为极径，y 轴为角度，FY0K'与标准点 FY0K($K \neq 0$，1)的偏差距离可以表示为两点之间的距离，从而比较出偏差最小的点 FY0K'。

2、此时，将该偏差最小的点近似认为在标准圆周上，选定标准点 FY01 与近似标准点 FY0K'为发射端，圆心飞机与圆周上其余 7 架飞机为接收端。

3、由于选定的发射端编号已知，故圆周上两架发射端飞机之间的弧长为定值，因为近似 FY0K'为标准点，所以可计算与 FY01 组成的圆心角近似于$\alpha = (\mathrm{K}-1)2\pi/9$，相当于通过最优点 FY0K'得到一个当前解集中最接近标准圆的解。

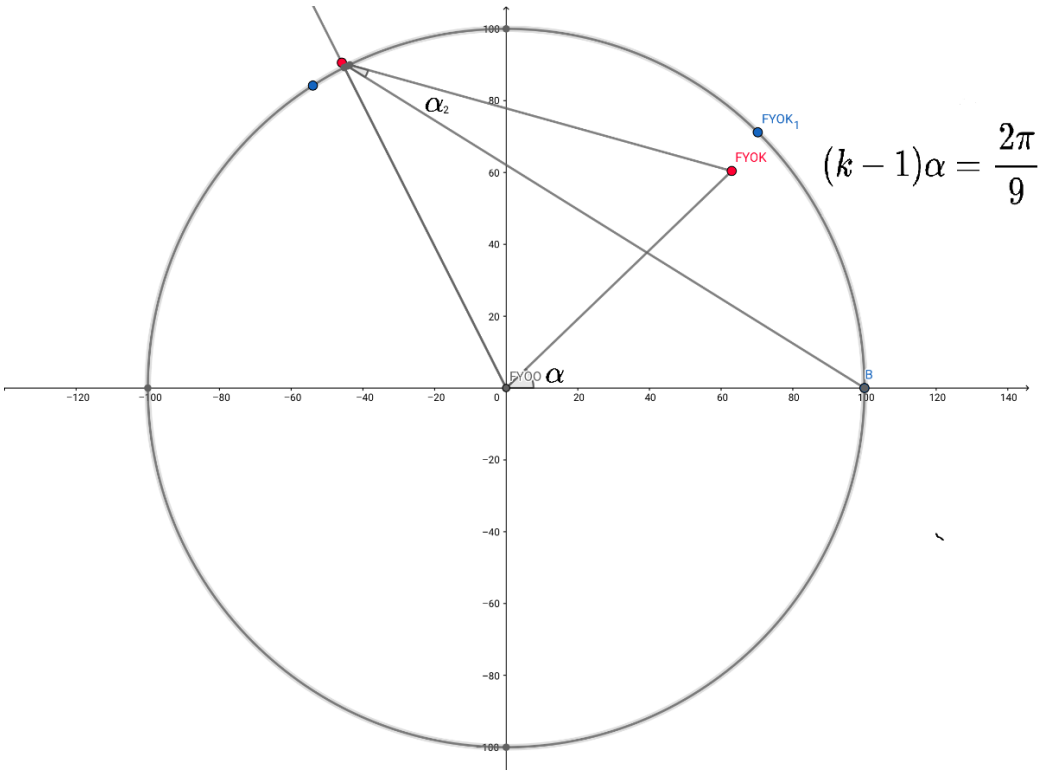


图 5.5 最接近标准圆的解

4、又因为无论哪一个接收信号的无人机得到的信号角都应该近似是$\alpha/2$，但实际接收无人机点位有偏差，且偏差程度不一，故而每台接收无人机所得信号角不同，由于无人机接收到的信号源编号是选定的。

5、已知的，故而无人机可知圆周角$\alpha/2$，然后无人机通过比较目前实际接收到的信号角与理论圆周角的大小，确定自身沿极径方向的移动方向，越接近圆心值越大，所以当信号角小于圆周角时，沿着极径向内移动，反之向外移动。

6、计算机算法实现则采用二分法使信号角无限逼近圆周角。依据此法将所有点偏移，完成第一次迭代，往后的每次迭代都选取极径上最优点，不考虑极角，往复迭代后，使得所有点无限逼近标准圆圆周。

7、完成极径的偏移调整后，再调整极角，仍然选择当前极角偏差最小的最优点 FY0K' $(K \neq 0，1)$与 FY01,FY00 组成发射端，当前可视为将某一点在经过 FY0K'的最接近标准圆的圆周上调整极角，无人机的调整依据角度为 FY0K'，FY01 信号角，保持该信号角不变则可保证无人机不脱离圆周。

8、将 FY00，FY01 信号角调整为$\gamma = (\pi - (N-1)\alpha)/2$，根据比较当前 FY00，FY01 信号角与 γ 的大小可判断出顺时针或逆时针移动方向，且要注意的是，在 FY0K'与 FY01 连线上方的点与下方的点移动规律相反。

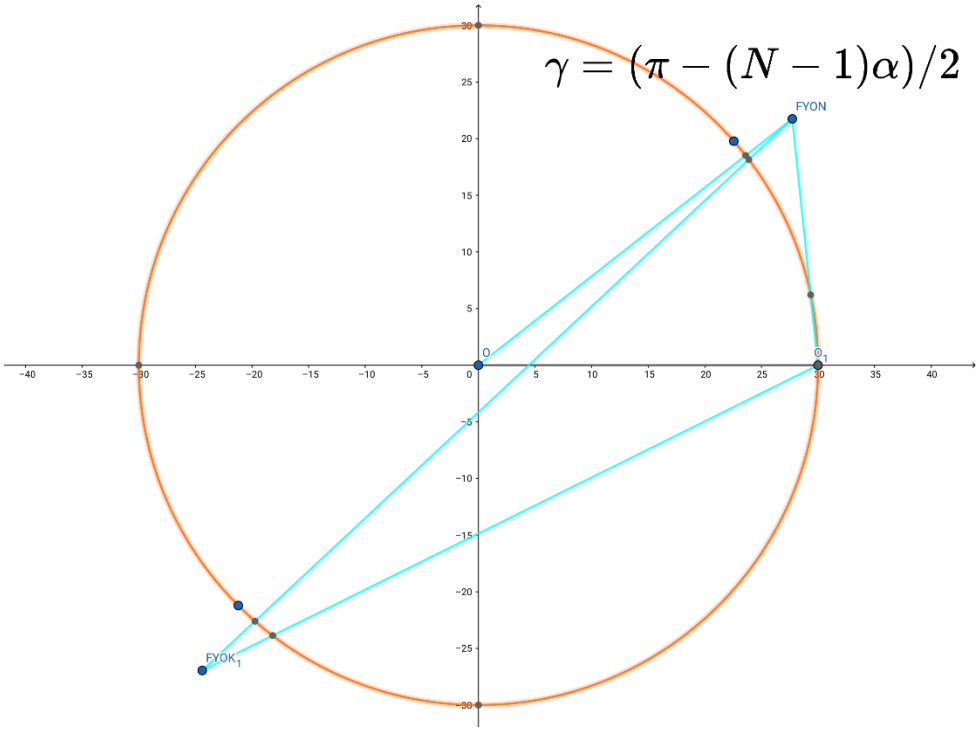


图 5.6 偏差角调整图

9、继续每次取极角偏差最小的点为最优点进行迭代，则可使所有圆周点无限逼近标准均匀分布。

5.2 问题二模型的建立与求解

5.2.1 问题分析

该问从实际飞行编队入手，调整为锥形队形，因相邻的两架无人机间距相等，故理想状态下应该是等边三角形，故我们需要尽可能的调整无人机的偏差，形成一个个标准的等边三角形。

5.2.2 模型的建立与求解

优化偏差值算法思路

因为理想状态下是标准的等边三角形，边长为 50，故而可求出没一个标准点的坐标，以及没一个标准点直接形成的角度。

首先选定 FY01 为领队无人机，过 FY01，FY11，FY15 做外接圆，为使得领队无人机尽可能的标准，我们固定 FY01 不动，移动 FY11 与 FY15，使得 FY01 收到 FY11，FY15 的信号角趋近 60°，具体移动方案如下:

。

我们知道，三角形中，大角对大边，假设 FY01 收到的信号角小于 60°，三角形中必定至少有一个角大于 60°，此时让 FY11 接收 FY01,FY15 的信号角，FY15 接收 FY01，FY11 的信号角，比较各自的信号角与 60° 的偏差，如果的信号角大于 60°，则应该向 x 轴正方向逼近，且逼近途中保持信号角保持不变，以确保无人机不会脱离圆周，直到信号角接近 60°。

此时就得到了一个由 FY01，FY11，FY15 组成的近似等边三角形，然后 FY01 仍

然是其中最标准的点，我们再比较 FY11 与 FY15 的偏差程度，得出最标准的点，此时使用这两个近似标准的点，分别去给深一层的三角形的三个顶点 FY08，FY05，FY09 发射信号，因为在理想状态下，这些信号角$\theta_1\theta_2\theta_3$可求，所以我们可以根据 FY08，FY05，FY09 所收到的信号角，与求出对应的θ比较，得出其中偏差最小的角，完成一轮迭代。

再将所得偏差最小的角以外的两个角中得出一个最小偏差角，继续以相同的方式向里层的三角形的三个顶点迭代，直到最深层为止。

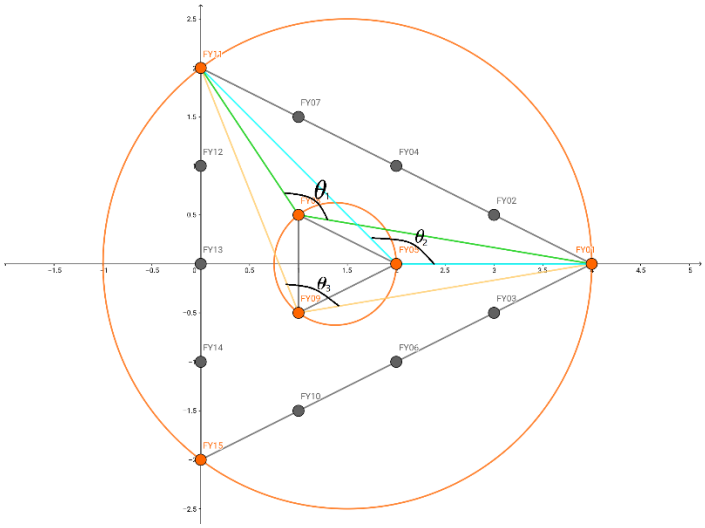


图 5.7 锥形编队模型图

此时由于我们队领队无人机的偏差精度未必满足我们的要求，且每层三角形边上的点还没调整，所以我们以最内层的三个点为标准，向外一层的所有顶点发射信号，用相同的方法将外层顶点差值精度优化。

再选定外层三角形的一条边的两个端点作为发射端，边上的点作为接收端，始终保持信号角为 180° 来调整点的位置，使其在调整过程中始终不脱离三角形的边，调整的方向就是内一层的一个顶点也发射信号。

进而可以计算预测出信号角的大小，与实际接受到的信号角大小比较，来确定沿着边的调整方向，如此调整完所有的边后，再以相同方式向外一层优化偏差值，直到所有点优化完毕。

六、 模型评价与改进

6.1 问题一模型评价与改进

(1)算法优缺点分析

优点:从极端角度出发，该算法可使得解无限逼近于精确答案。

缺点:当无人机阵列规模庞大时，发射信号次数大，难满足电磁静默，故只能在如题这样的小规模阵列下使用。

(2)算法的误差分析与控制

算法的误差较大成分来源于随**迭代**深度生长的信号发射次数，而迭代的深度取决于需要精确到小数点后多少位，在满足数据精确的同时，最理想的情况下发射的信号次数为 29 次，而根据实际情况考虑，无人机阵列紧密，信号失真率低，且完成一次发射信号所需时间极短，纵使几百次的信号发射也是在一瞬间完成，故而可将误差控制在可接受的范围内。

3.思考最优解的算法方向

可以尝试**粒子群优化算法**根据题目情况设计出粒子群算法的衍生算法，因题意不考虑无人机飞行偏移过程中的速度，故而设所有无人机的速度为单位一，以 FY00 飞机为原点建立极坐标系，每一架飞机的自主最优解与集合最优解可定义算法映射为角度的偏移方向与极径的变化方向，则可以依此建立数学模型求解。

(1)算法的最优解分析

由于该算法使得每架无人机同时偏向两个最优解，故而区别于算法 2 不同的是，算法 2 中极角的偏差与极径的偏差分开调整，故而算法效率与信号量级明显劣与该算法。该算法选中几架发射端便可使其他接收端同时偏向标准点，在几次迭代后便可得解。在该算法中，可实现接收端同时移动，大大提升了算法的高效性。

6.2 问题二模型评价与改进

算法优缺点分析

优点:该算法所得结果准确度较高，能二次优化数据，且可用于规模更大的锥形无人机阵列。

缺点:当无人机阵列规模巨大时，该算法所要发射的信号量巨大。

七、 参考文献

[1]武瑞宏.空中三角测量的再认识[J].测绘通报,2002,(03):8-10.

[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]周文雅,李哲,许勇,杨峰,贾涛.基于双目视觉的无人机编队相对定位算法[J].宇航学报,2022,43(01):122-130.

[4]关欣,关欣,陶李,衣晓.多站协同定位的定位精度研究[J].计算机与数字工程,2016,44(05):829-834.

[5]张鲲,沈重,王海丰,李壮,高倩,李涵雯.海上侦察船的纯方位无源定位技术研究[J].舰船科学技术,2018,40(02):19-21.

附录

附录一：第一问第一小问**C#程序**

```
    /// <summary>
            /// 获得三点的外接圆
            /// </summary>
            /// <returns></returns>
            /// <summary>
            /// <para>二维：已知圆上三点，求圆心坐标</para>
            /// <para>三个点不在同一直线上</para>
            /// </summary>
            /// <param name="P1">点 1</param>
            /// <param name="P2">点 2</param>
            /// <param name="P3">点 3</param>
            /// <returns></returns>
            public static KeyValuePair<float,float>
GetThreePointCircle(PointF P1, PointF P2, PointF P3)
            {
                double a13 = P1.X - P3.X;
                double a13_ = P1.X + P3.X;
                double b13 = P1.Y - P3.Y;
                double b13_ = P1.Y + P3.Y;
                double a12 = P1.X - P2.X;
                double a12_ = P1.X + P2.X;

                double b12 = P1.Y - P2.Y;
                double b12_ = P1.Y + P2.Y;

                double a12b12_2 = a12 * a12_ + b12 * b12_;
                double a13b13_2 = a13 * a13_ + b13 * b13_;

                double a13b12 = 2 * a13 * b12;
                double a12b13 = 2 * a12 * b13;

                if (a12b13 - a13b12 == 0) return new KeyValuePair<float,
float>((P2.X + P1.X) / 2, (P2.Y + P1.Y) / 2);
                double af = a12b13 - a13b12;
                double bf = a13b12 - a12b13;
                double az = b13 * a12b12_2 - b12 * a13b13_2;
                double bz = a13 * a12b12_2 - a12 * a13b13_2;
                double a = az / af;
                double b = bz / bf;
```

```
            return new KeyValuePair<float, float>((float)a, (float)b);
        }
```

附录二：第一问第三小问**C#程序**

```
        /// <summary>
        /// 数据归一化取得最短距离
        /// </summary>
        /// <param name="msg"></param>
        /// <returns></returns>
        public static float GetMinLength(KeyValuePair<float,float>[]
msg)
        {
            float minLength = 1000;
            for(int i = 2; i < 10; i++)
            {
                double tempLength =
Math.Sqrt(Math.Pow(Math.Abs(msg[i].Key - 100),2) +
Math.Pow(Math.Abs(msg[i].Value - (i - 1) * 40),2));
                if(tempLength!=0 && tempLength < minLength)
                {
                    minLength = (float)tempLength;
                }
            }
            return minLength;
        }

        /// <summary>
        /// 得到三个点组成的角
        /// </summary>
        /// <param name="p1">点 1</param>
        /// <param name="p2">点 2</param>
        /// <param name="endp">顶点</param>
        /// <returns></returns>
        public static float GetThreePointAngle(KeyValuePair<float,float>
p1,KeyValuePair<float,float> p2,KeyValuePair<float,float> endp)
        {
            float a = GetTwoPointDistance(p1, endp);
            float b = GetTwoPointDistance(p2, endp);
            float c = GetTwoPointDistance(p1, p2);
            float Angle = (float)Math.Acos((Math.Pow(a, 2) + Math.Pow(b,
2) - Math.Pow(c, 2)) / (2 * a * b));
```

```
            return Angle;
        }
```

附录三：第一问第三小问**C#程序**

```
        /// <summary>
        /// 得到三个点组成的角
        /// </summary>
        /// <param name="p1">点 1</param>
        /// <param name="p2">点 2</param>
        /// <param name="endp">顶点</param>
        /// <returns></returns>
        public static float GetThreePointAngle(KeyValuePair<float,float>
p1,KeyValuePair<float,float> p2,KeyValuePair<float,float> endp)
        {
            float a = GetTwoPointDistance(p1, endp);
            float b = GetTwoPointDistance(p2, endp);
            float c = GetTwoPointDistance(p1, p2);
            float Angle = (float)Math.Acos((Math.Pow(a, 2) + Math.Pow(b,
2) - Math.Pow(c, 2)) / (2 * a * b));
            return Angle;
        }
```

附录四：第一问第三小问**C#程序**

```
    /// <summary>
        /// 初始化已知点的数据信息，返回一个键值对数组
        /// </summary>
        /// <returns></returns>
        public static KeyValuePair<float,float>[] InitErrorMsg()
        {
            KeyValuePair<float,float>[] msg = new KeyValuePair<float,
float>[10];
            msg[0] = new KeyValuePair<float, float>(0, 0);
            msg[1] = new KeyValuePair<float, float>(100, 0);
            msg[2] = new KeyValuePair<float, float>(98, 40.10f);
            msg[3] = new KeyValuePair<float, float>(112, 80.21f);
            msg[4] = new KeyValuePair<float, float>(105, 119.75f);
            msg[5] = new KeyValuePair<float, float>(98, 159.86f);
            msg[6] = new KeyValuePair<float, float>(112, 199.96f);
            msg[7] = new KeyValuePair<float, float>(105, 240.07f);
            msg[8] = new KeyValuePair<float, float>(98, 280.17f);
```

```
            msg[9] = new KeyValuePair<float, float>(112, 320.28f);
            return msg;
        }
```

附录五：其他程序

```
    /// <summary>
        /// 把每个点转化为直角坐标系上的点
        /// </summary>
        /// <param name="msg"></param>
        /// <returns></returns>
        public static KeyValuePair<float,float>[]
InitErrorPoint(KeyValuePair<float,float>[] msg)
        {
            KeyValuePair<float, float>[] pointMsg = new
KeyValuePair<float, float>[msg.Length];
            pointMsg[0] = msg[0];
            //pointMsg[1] = msg[1];
            for(int i = 1; i < msg.Length; i++)
            {
                float x = (float)(msg[i].Key *
Math.Cos((msg[i].Value*(Math.PI))/180));
                float y = (float)(msg[i].Key *
Math.Sin((msg[i].Value*(Math.PI))/180));
                pointMsg[i] = new KeyValuePair<float, float>(x,y);
            }
            return pointMsg;
        }

     /// <summary>
        /// 得到两点之间的距离
        /// </summary>
        /// <param name="p1"></param>
        /// <param name="p2"></param>
        /// <returns></returns>
        public static float
GetTwoPointDistance(KeyValuePair<float,float> p1,KeyValuePair<float,float>
p2)
        {
            return (float)Math.Sqrt(Math.Pow(p1.Key - p2.Key, 2) +
Math.Pow(p1.Value - p2.Value, 2));
        }
```