基于最小二乘法和迭代法的无人机飞行编队纯方位无源定位与调整的模型

摘要

本文主要研究无人机编队飞行纯方位无源定位问题，建立了基于**最小二乘法**和**迭代法**的无人机飞行编队纯方位无源定位与调整模型。

对于问题一 (1)，对单个无人机建立**基于最小二乘法的最优定位模型**，在实际情况中，无人机获得的角度信息可能不准确，因此采用**最小二乘法**进行最优定位。以发送信号的无人机的坐标和提供的夹角信息构建方程组，再利用**最小二乘法**，求解出位置略有偏差的无人机的最优坐标。

对于问题一 (2)，对于单个无人机，建立**基于二维角度坐标差值比较的编号推算模型**。由题目信息知，无人机的偏差值较小，因此可以由该夹角与理想夹角的差值确定无人机的编号。由于每次获得的夹角有 2 个，故将得到的角度转化为二维坐标形式，并与理想情况下的所有坐标进行**加权距离**的计算，最终找到最小加权距离对应编号，该编号即为未知编号无人机的编号，因而问题转化为问题一 (1)，利用相同模型求解。故对于问题一 (2)，**除 FY00 和 FY01 外，需要 1 架无人机发射信号，即可实现无人机的有效定位。**

对于问题一 (3)，首先建立了单个无人机的**矢量位移方法**。即无人机可以在已知有 3 架无偏差的无人机发送信号的情况下，在坐标系中进行方向确定、大小确定的矢量位移；据此确立了**基于迭代法的多个无人机定位模型**。若所有无人机都能最终准确得知自己当前的坐标，则根据矢量位移方法，所有无人机都能调整至理想位置。因此该问题划归为多无人机**在最多只有两架无人机已知位置准确的情况下的定位问题**。在该问题中，采用**迭代法**对各个无人机的坐标进行多次迭代求解，最终无人机的坐标会收敛至实际坐标。

对于问题二，定位某个有偏差的无人机的位置的方法与问题一 (1) 的模型相同，但相比于问题一 (3)，会出现 3 架无人机处于同一直线而无法求解的情况。因此先利用问题一 (3) 中的迭代模型确定平行四边形顶点处的无人机真实坐标再利用问题一 (1) 中的定位模型进行**递推**，最终算得所有无人机的坐标。

关键字：纯方位无源定位；最小二乘法；二维角度坐标差值比较；迭代法；递推

1　问题背景与重述

1.1　问题背景

在无人机进行集群遂行编队飞行时，为了避免外界干扰，通常会选择较少向外部发射电磁波信号，即尽可能保持电磁静默。因此，为保持编队队形，本题目要求采用纯方位无源定位的方法调整无人机的位置。具体方法如下：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。接收信号的无人机所接收到的方向信息约定为：该无人机与任意两架发射信号无人机连线之间的夹角。如图 1 所示，编号为 FY01、FY02 及 FY03 的无人机发射信号，编号为 FY04 的无人机接收到的方向信息是 $\alpha 1$，$\alpha 2$ 和 $\alpha 3$。

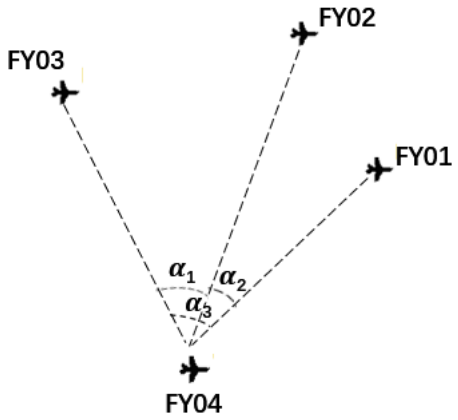

图 1　锥形无人机编队示意图

1.2　问题重述

1.2.1　问题一

如图 2 所示，编队由 10 架无人机组成，形成圆形编队，其中 9 架无人机（编号 FY01-FY09）均匀分布在某一圆周上，另 1 架无人机（编号 FY00）位于圆心。无人机基于自身感知的高度信息，均保持在同一个高度上飞行。

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

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

(3) 按编队要求，1 架无人机位于圆心，另 9 架无人机均匀分布在半径为 100m 的圆周上。当初始时刻无人机的位置略有偏差时，请给出合理的无人机位置调整方案，即通

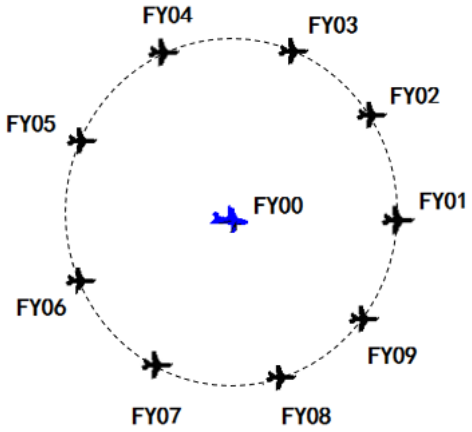

图 2　圆形无人机编队示意图

过多次调整，每次选择编号为 FY00 的无人机和圆周上最多 3 架无人机遂行发射信号，其余无人机根据接收到的方向信息，调整到理想位置（每次调整的时间忽略不计），使得 9 架无人机最终均匀分布在某个圆周上。仅根据接收到的方向信息来调整无人机的位置，给出具体的调整方案。

1.2.2　问题二

实际飞行中，无人机集群也可以是其他编队队形，例如锥形编队队形，如图 3 所示，直线上相邻两架无人机的间距相等，如 50m。仍考虑纯方位无源定位的情形，设计无人机位置调整方案。

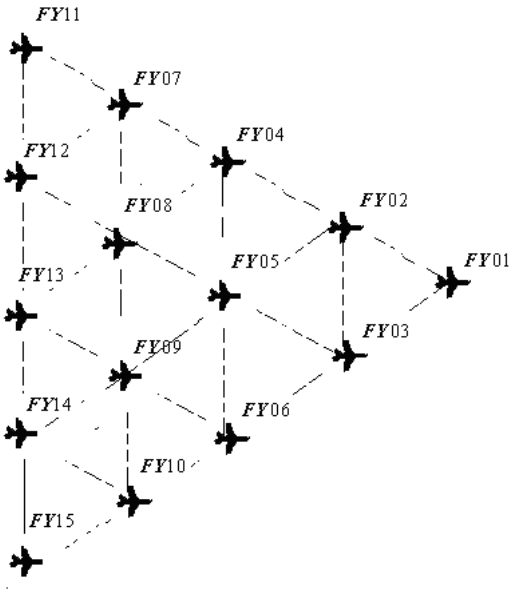

图 3　锥形无人机编队示意图

2 模型假设

(1) 假设无人机在接受角度的信息时，能够得知与自身形成该夹角的两架无人机的身份信息。例如：FY00 与 FY01 向 FY05 发送信息，则 FY05 得知 FY00 与 FY01 与自身构成的夹角 α；若 FY00 与另一架未知编号的无人机向 FY05 发送信息，则 FY05 得知 FY00 与一架未知编号的无人机与自身构成的夹角 β。

(2) 假设在无人机调整方位的过程中，FY00 的位置恒为定点，即以 FY00 为参考系。

(3) 假设无人机可以根据自身当前的朝向来调整至任意方向并移动任意距离，例如，无人机可以在某一时刻向右旋转 60°，再向前行进 20m。

(4) 假设问题一中无人机略有偏差的范围为极坐标内 $|\Delta r| \leq 12\text{m}, |\alpha| \leq 1°$，假设问题二中无人机略有偏差的范围为直角坐标内 $|\Delta x| \leq 5\text{m}, |\text{y}| \leq 5\text{m}$

(5) 在问题一的 (3) 中，假设 FY00 与 FY01 均处于理想位置；在问题二中，假设 FY01 与 FY02 均处于理想位置。

3 符号说明

符号	意义	单位
R	编队圆周的半径	米 (m)
l_{AB}	圆上 AB 两点的弦长	米 (m)
$r_{AB,\alpha}$	以 AB 为弦，圆周角为 α 的圆的半径	米 (m)
$\alpha_{a,b,c}$	无人机 FY0a 收到的来自 FY0b 和 FY0c 的角度信息	度 (°)
D_{mn}	角度二维坐标中 FY0m 和 FY0n 两点的加权距离	米 (m)
X_m	编号为 m 的无人机所处直角坐标系中的理想 x 值	米 (m)
x_{m_p}	编号为 m 的无人机经过 p 次迭代后的坐标 x 值	米 (m)
y_{m_p}	编号为 m 的无人机经过 p 次迭代后的坐标 y 值	米 (m)
Y_m	编号为 m 的无人机所处直角坐标系中的理想 y 值	米 (m)
Δw	计算坐标时所产生的偏差值	米 (m)

4 问题分析

4.1 问题一分析

问题一主要在于如何根据其他位置准确的无人机提供的信息确定某架具有编号的无人机的准确位置。

对于问题一 (1)，对单个无人机建立基于最小二乘法的最优定位模型，在实际情况中，无人机获得的角度信息可能不准确，因此采用最小二乘法进行最优定位。以发送信号的无人机的坐标和提供的夹角信息构建方程组，再利用最小二乘法，求解出位置略有偏差的无人机的最优坐标。

对于问题一 (2)，对于单个无人机，建立基于二维角度坐标差值比较的编号推算模型。由题目信息知，无人机的偏差值较小，因此可以由该夹角与理想夹角的差值确定无人机的编号。由于每次获得的夹角有 2 个，故将得到的角度转化为二维坐标形式，并与理想情况下的所有坐标进行加权距离的计算，最终找到最小加权距离对应编号，该编号即为未知编号无人机的编号，因而问题转化为问题一 (1)，利用相同模型求解。故对于问题一 (2)，除 FY00 和 FY01 外，需要 1 架无人机发射信号，即可实现无人机的有效定位。

对于问题一 (3)，首先建立了单个无人机的矢量位移方法。即无人机可以在已知有 3 架无偏差的无人机发送信号的情况下，在坐标系中进行方向确定、大小确定的矢量位移；据此确立了基于迭代法的多个无人机定位模型。若所有无人机都能最终准确得知自己当前的坐标，则根据矢量位移方法，所有无人机都能调整至理想位置。因此该问题划归为多无人机在最多只有两架无人机已知位置准确的情况下的定位问题。在该问题中，采用迭代法对各个无人机的坐标进行多次迭代求解，最终无人机的坐标会收敛至实际坐标。

4.2 问题二分析

对于问题二，定位某个有偏差的无人机的位置的方法与问题一 (1) 的模型相同，相比于问题一 (3)，会出现 3 架无人机处于同一直线的情况。假定一开始 FY01 和 FY02 的相对位置准确，每次计算新的近似坐标时，令作为基准的两台无人机和待定位的两台无人机分别位于同一平行四边形的相对的两边的顶点上。再利用问题一 (1) 中的定位模型进行递推，最终算得所有无人机的坐标。

5 模型建立与求解

5.1 问题一模型的建立

5.1.1 问题一 (1)：基于最小二乘法的单个无人机精确定位模型

在理想情况下，利用角度信息获得方程求解出交点即为无人机坐标。但在实际情况中，无人机获得的角度信息可能不准确，因此采用最小二乘法进行进一步最优定位。以发送信号的无人机的坐标和形成的夹角得到线性方程组，再利用最小二乘法，可以求解出位置略有偏差的无人机的最优坐标。

(1) 以发送信号的无人机的坐标和形成的夹角得到线性方程组

设编队圆周的半径为 R，已知 FY00 位置理想且发送信号，将 FY00 位置设为原点，建立平面直角坐标系，将另外 2 架无人机重命名为 FY0m 和 FY0n，将需要求解坐标的有偏差的无人机重命名为 FY0x，其中 FY0m 置设于 x 轴上，即 (x_1, y_1)=$(R, 0)$，FY0n 坐标可设为 (x_2, y_2)，若转化为极坐标，则为 $(R, 0)$ 和 (R, θ)。

现有 3 架位置准确且可发出信号的无人机，在平面直角坐标系中分别为 FY00(0,0)，FY0m(x_1, y_1)，FY0n(x_2, y_2)，FY00 与 FY0m 提供的角度信息为 $\alpha_{x,0,m}$，FY00 与 FY0n 提供的角度信息为 $\alpha_{x,0,n}$，FY0m 与 FY0n 提供的角度信息为 $\alpha_{x,m,n}$。选取其中任意 2 架无人机，结合这 2 架无人机的理想位置坐标和提供的角度信息可以获得 2 个圆方程，略有偏差的目标无人机 FY0x 位于其中 1 个圆上。

具体求解方法如下：设坐标系中有点 A(x_1, y_1) 和 B(x_2, y_2)，设 AB 距离为 l_{AB}，则有

$$l_{AB} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

以 AB 为弦，圆周角为 α 的圆的半径 $r_{AB,\alpha}$ 为

$$r_{AB,\alpha} = \frac{l_{AB}}{2sin\alpha} \tag{2}$$

圆心到 AB 的距离 d 为

$$d = \sqrt{r_{AB,\alpha}^2 - \left(\frac{l_{AB}}{2}\right)^2} \tag{3}$$

根据几何关系求得圆心坐标为

$$\begin{cases} x_0 = \dfrac{x_1 + x_2}{2} \pm \dfrac{d}{l_{AB}}(y_2 - y_1) \\ y_0 = \dfrac{y_1 + y_2}{2} \pm \dfrac{d}{l_{AB}}(x_1 - x_2) \end{cases} \tag{4}$$

圆心方程为

$$(x - x_0)^2 + (y - y_0)^2 = r_{AB,\alpha}^2 \tag{5}$$

同理，另取不同组合的 2 架无人机，可以再次获得 2 个圆方程。3 个点所产生的圆方程共有 3 组，6 个。对于每组圆方程，偏差无人机 FY0x 位于其中 1 个圆上，故应当利用一定数学方法舍去另一个无效圆。

舍去无效圆的具体方法如下：对于一组圆，分别计算 2 个圆的圆心与偏差无人机的理想位置坐标的距离与 $r_{AB,\alpha}$ 之差的绝对值，差值较小者为有效圆的交点，从而找出有效圆，舍去无效圆。

最终，得到 3 个有效圆，即每个圆上都存在 2 个基准点，以及 FY0x 的坐标点。

(2) 利用最小二乘法求解最优近似坐标

在理想情况下，在 3 个有效圆中任取 2 个圆，求得 2 个交点坐标，除基准点以外的 1 个点为 FY0x 的实际坐标。但在实际情况中，无人机获得的角度信息可能不准确，因此选择不同无人机组合获得的圆方程求解出的坐标可能并不一致。因此，考虑最坏情况，即获得的 3 个角度均有一定的偏差，则最终由不同无人机组合获得的坐标共有 3 个，此时可转换方法，利用 3 个有效圆方程彼此相减消去二次项得到线性方程组：

$$\begin{cases} 2(x_1-x_2)x+2(y_1-y_2)y= & (x_1^2+y_1^2-r_1^2)-(x_2^2+y_2^2-r_2^2) \\ 2(x_0-x_2)x+2(y_0-y_2)y= & (x_0^2+y_0^2-r_0^2)-(x_2^2+y_2^2-r_2^2) \\ 2(x_1-x_0)x+2(y_1-y_0)y= & (x_1^2+y_1^2-r_1^2)-(x_0^2+y_0^2-r_0^2) \end{cases} \tag{6}$$

其中 (x_0,y_0)，(x_1,y_1)，(x_2,y_2) 分别为 3 个有效圆的圆心坐标，r_0，r_1，r_2 分别为 3 个有效圆的半径。再将方程组写成矩阵形式：

$$A=\begin{bmatrix} 2(x_1-x_2) & 2(y_1-y_2) \\ 2(x_0-x_2) & 2(y_0-y_2) \\ 2(x_1-x_0) & 2(y_1-y_0) \end{bmatrix}, \mathbf{b}=\begin{bmatrix} (x_1^2+y_1^2-r_1^2)-(x_2^2+y_2^2-r_2^2) \\ (x_0^2+y_0^2-r_0^2)-(x_2^2+y_2^2-r_2^2) \\ (x_1^2+y_1^2-r_1^2)-(x_0^2+y_0^2-r_0^2) \end{bmatrix} \tag{7}$$

原方程为

$$A\mathbf{x}=\mathbf{b}. \tag{8}$$

写出其正规方程：

$$A^TA\mathbf{x}=A^T\mathbf{b}. \tag{9}$$

求解出其得出最优近似坐标 (x,y)，此题得解。经检验，求解坐标与实际坐标误差值小于 0.01%，检验代码存于附件 problem1a.py。

5.1.2 问题一 (2)：基于二维坐标角度差值比较的单个无人机编号推算模型

问题 (2) 与问题 (1) 不同的是，其中 1 架无人机的编号未知，则问题 (2) 首要考虑的目标是将该无人机的编号确定，继而能够将问题 (2) 转化为问题 (1) 进行求解。

由题目信息知，无人机的偏差值较小，结合问题 (3) 中表 1 数据，可以认定角度的偏差仅在 1° 以内，半径值的偏差在 12m 以内，因此可以由该夹角与理想夹角的差值确定无人机的编号。

现设需要定位的偏差无人机为 FY0x，可发射信号的 3 架无人机分别为 FY00，FY01 和 FY0m，其中 FY00，FY01 编号已知，而 FY0m 编号未知。

设 FY00 和 FY0m 提供的角度信息为 $\alpha_{x,0,m}$，FY01 和 FY0m 提供的角度信息为 $\alpha_{x,1,m}$。以这两个角度信息分别作为 x 和 y 变量建立平面直角坐标系，设 FY0m 的坐标为 $(\alpha_{x,0,m},\ \alpha_{x,1,m})$。另用理想位置坐标得出除 FY00，FY01 和 FY0x 以外所有无人机对应的坐标。在该平面直角坐标系中，计算 FY0m 与其他各点坐标（如 FY0p($\alpha_{x,0,p}$, $\alpha_{x,1,p}$)）的加权距离（x 坐标的权重高于 y 坐标的权重）：

$$D_{mp} = \sqrt{(\alpha_{x,0,m} - \alpha_{x,0,p})^4 + (\alpha_{x,1,m} - \alpha_{x,1,p})^2} \tag{10}$$

并找到最短加权距离对应的坐标，该坐标对应的无人机编号即为 FY0m 的编号。其中，加权距离函数中指数系数 4 和 2 均为进行数据统计后给出的最优结果。在该加权距离公式下，且误差范围在 $|\Delta r| \leq 12m, \Delta\alpha| \leq 1°$ 之内时，准确率为 100.00%, 误差范围在 $|\Delta r| \leq 13m, |\Delta\alpha| \leq 2°$ 之内时，准确率为 99.80%，在误差允许范围内。在得知 FY0m 的具体编号后，利用问题 (1) 中的模型求解。检验代码存于附件 problem1b.py。

由此得出结论，若发射信号的无人机位置无偏差，除 FY00 和 FY01 外，**还需要 1 架无人机发射信号，才能实现无人机的有效定位**。

5.1.3 问题一 (3)：基于迭代法的多个无人机定位模型

在问题一 (3) 中，单个无人机的坐标无法通过一次操作精确定位，因此考虑使用多次迭代法求出所有无人机的近似坐标。结合无人机的矢量位移办法，使所有无人机向自身的理想位置进行矢量位移，最终所有无人机被调整至理想位置。

(1) 无人机的矢量位移方法

根据问题一 (1)，现已确定在 3 架无人机位置无偏差且可以发送信号的情况下，任意 1 架无人机的真实坐标可以被定位。现在预先设定的平面直角坐标系中，可求得 1 架无人机当前的坐标 (x_0, y_0)，令无人机向任意方向位移任意距离，再次求得无人机坐标 (x_1, y_1)，根据 2 个坐标可以确定无人机进行位移的方向对应在坐标系中的矢量方向，并且根据坐标距离可以得知无人机位移的实际距离。故无人机可以在已确定在 3 架无人机位置无偏差且可以发送信号的情况下，进行任意方向、任意大小的矢量位移。

(2) 利用迭代法求解所有无人机的近似坐标

在问题 (3) 中，若所有无人机都能最终准确得知自己当前的坐标，则根据已推导出的无人机矢量位移办法，所有无人机都能调整至理想位置。因此此问题划归为多无人机在最多只有两架无人机已知位置准确的情况下的定位问题。

已知 FY00、FY01 位置无偏差，且其余无人机离理想位置偏差较小，则可将其余无人机两两分为一组，例如 $\{2,6\}\{3,7\}\{4,8\}\{5,9\}$，预先假设无人机处于理想位置，即所有无人机的计算预设初始值

$$(x_m, y_m) = (X_m, Y_m), m \in \{0,1,2,3,4,5,6,7,8,9\} \tag{11}$$

在每一组内，预先假设无人机处于理想位置 $(x_{m_0}, y_{m_0}) = (X_m, Y_m)$ 和 $(x_{n_0}, y_{n_0}) = (X_n, Y_n)$，利用问题一 (1) 的中基于最小二乘法的单个无人机精确定位模型，计算并更新 2 个无人机的位置坐标 (x_{m_1}, y_{m_1}) 和 (x_{n_1}, y_{n_1})。多次迭代计算，最终使两者的坐标都收敛到各自真实的坐标。

以 $\{2, 6\}$ 为例。两者坐标的初始值都预先设为其理想坐标 (X_2, Y_2) (X_6, Y_6)。首先令 FY00，FY01 与 FY06 给 FY02 发送角度信息 $\alpha_{2,0,1}$、$\alpha_{2,0,6}$、$\alpha_{2,1,6}$，据此计算出 FY02 新的近似位置 (x_{2_1}, x_{2_1})。再令 FY00，FY01 与 FY02 给 FY06 发送角度信息 $\alpha_{6,0,1}$、$\alpha_{6,0,2}$、$\alpha_{6,1,2}$，据此计算出 FY06 新的近似位置 (x_{6_1}, x_{6_1})。如此循环，在两者坐标与前一次迭代的偏差 $|\Delta w| < 0.01\text{m}$ 时停止。每次计算新的近似坐标时，FY02 与 FY06 使用的都是当前最新的近似坐标值。4 组无人机都分别进行这一过程，最终得到所有无人机的准确坐标。

5.2 问题一模型的求解

5.2.1 无人机的初始位置与理想位置坐标

无人机初始位置与理想位置的直角坐标与极坐标计算如表 1，结果以 2 位小数形式表示。

表 1 无人机初始位置与理想位置的直角坐标与极坐标

无人机编号	真实极坐标	理想极坐标	真实直角坐标	理想直角坐标
00	(0.00, 0.00)	(0.00, 0.00)	(0.00, 0.00)	(0.00, 0.00)
01	(100.00, 0.00)	(100.00, 0.00)	(100.00, 0.00)	(100.00, 0.00)
02	(98.00, 40.10)	(100.00, 40.00)	(74.96, 63.12)	(76.60, 64.28)
03	(112.00, 80.21)	(100.00, 80.00)	(19.04, 110.37)	(17.36, 98.48)
04	(105.00, 119.75)	(100.00, 120.00)	(-52.10, 91.16)	(-50.00, 86.60)
05	(98.00, 159.86)	(100.00, 160.00)	(-92.01, 33.74)	(-93.97, 34.20)
06	(112.00, 199.96)	(100.00, 200.00)	(-105.27, -38.23)	(-93.97, -34.20)
07	(105.00, 240.17)	(100.00, 240.00)	(-52.39, -91.00)	(-50.00, -86.60)
08	(98.00, 280.17)	(100.00, 280.00)	(17.30, -96.46)	(17.36, -98.48)
09	(112.00, 320.28)	(100.00, 320.00)	(86.15, -71.57)	(76.60, -64.28)

5.2.2 迭代过程求解

根据代码仿真模拟结果，将每个无人机的各个坐标收敛到误差 $|\Delta w| < 0.01\text{m}$ 仅需 7 次迭代；误差 $|\Delta w| < 0.001\text{m}$ 仅需 10 次迭代，收敛情况较好。

表 2 问题一模型进行多次迭代后全部无人机坐标的近似结果

无人机编号	7 次迭代后近似直角坐标	10 次迭代后近似直角坐标
02	(74.9623, 63.1241)	(74.9623, 63.1241)
03	(19.0424, 110.3681)	(19.0439, 110.3689)
04	(-52.1027, 91.1630)	(-52.1027, 91.1612)
05	(-92.0077, 33.7429)	(-92.0077, 33.7429)
06	(-105.2723, -38.2328)	(-105.2723, -38.2328)
07	(-52.3888, -90.9988)	(-52.3889, -90.9971)
08	(17.3031, -96.4598)	(17.3037, -96.4602)
09	(86.1478, -71.5721)	(86.1478, -71.5721)

5.3 问题二模型的建立

对于问题二，定位某个有偏差的无人机的位置的方法与问题一 (1) 的模型相同，相比于问题一 (3)，会出现 3 架无人机处于同一直线的情况。假定一开始 FY01 和 FY02 的相对位置准确，每次计算新的近似坐标时，令作为基准的两台无人机和待定位的两台无人机分别位于同一平行四边形的相对的两边的顶点上。再利用问题一 (1) 中的定位模型进行递推，最终算得所有无人机的坐标。

5.4 问题二模型的求解

如表 3，假设位置正确的两架无人机是 FY01 和 FY02，则在 FY03 与 FY05 的真实位置与理想位置的误差小于等于 5 米的情况下进行代码随机数模拟仿真，将每个无人机的各个坐标收敛到误差 $|\Delta w| < 0.01$m 仅需 5 次迭代；误差 $|\Delta w| < 0.001$m 仅需 7 次迭代，收敛情况较好。最终全部结果如表 4：

表 3 问题二模型进行多次迭代后的 FY03 和 FY05 坐标近似结果

编号	初始值	迭代 5 次后结果	迭代 7 次后结果	真实坐标
03	(25.000, 43.3013)	(27.4912, 47.6321)	(27.4996, 47.6314)	(27.5000, 47.6314)
05	(74.9978, 43.3000)	(71.0141, 40.9988)	(71.0141, 40.9999)	(71.0141, 41.0000)

表 4 问题二模型进行多次迭代和递推后全部无人机坐标的近似结果

编号	初始值（理想值）	迭代后结果	真实坐标
01	(0,0)	(0,0)	(0,0)
02	(50.0000,0)	(50.0000,0)	(50.0000,0)
03	(25.0000,43.3013)	(27.8318,47.4383)	(27.8319,47.4383)
04	(100.0000,0)	(105.0000,-0.0014)	(105.0000,0)
05	(74.9978,43.3000)	(70.7261,41.4948)	(70.7261,41.4948)
06	(50,86.6025)	(51.4589,88.0681)	(51.4618,88.0630)
07	(150.0000,0)	(144.9988,0.0081)	(145.0000,0)
08	(125.0167,43.2909)	(124.0400,45.1461)	(124.0394,45.1467)
09	(99.9994,86.6222)	(99.6212,86.6023)	(99.6217,86.5998)
10	(75.0000,129.9038)	(76.0064,131.6302)	(76.0000,131.6359)
11	(200.0000,0)	(203.9847,2.4922)	(203.9848,2.4923)
12	(175.0008,43.3083)	(179.4991,44.7614)	(179.5047,44.7556)
13	(149.9956,86.6000)	(147.8144,83.9695)	(147.8141,83.9071)
14	(125.0065,129.9010)	(127.8193,132.3509)	(127.0065,132.3585)
15	(100,173.2051)	(99.0949,170.2621)	(99.0949,170.2621)

6 模型的补充说明

在问题一 (3) 与问题二中，我们都假定预先已有 2 架无人机的相对距离等于它们在理想情况下的相对距离。在问题一 (3) 中，结合表格中给出的数据，这 2 架无人机分别为 FY00 与 FY01；在问题二中，这 2 架无人机分别为 FY01 和 FY02。但从一般意义上而言，该假设对于模型的求解不具有必要性，即在该假设不存在的情况下，无人机的准确定位依然可以实现。在每架无人机距理想位置偏差较小的情况下，可假设其中 2 架的距离为理想距离，例如 100 米，并以它们为基准建立平面直角坐标系，再根据问题一 (1) 中的定位模型计算出所有无人机的坐标。再利用问题一 (3) 中的无人机矢量移动方法，令任意一台无人机往任意方向移动单位长度 1m，并计算出它当前的坐标。假设其前后坐标分别为 (x_1, y_1) 与 (x_2, y_2)，则该无人机在当前坐标系下的位移大小为

$$l = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

因此有

$$l = 1\text{m}.$$

由此，将所有无人机的每个坐标都乘以

$$k = \frac{1}{l},$$

可以得到所有无人机以 1m 为标准单位的准确坐标。在实际情况中，可多次测算 k 值并取平均以减小误差。

7 模型评价与改进

7.1 模型的优点

(1) 问题一 (1) 中基于最小二乘法的单个无人机最优定位模型和问题一 (2) 中基于二维角度坐标差值比较的单个无人机编号推算模型，均建立在严格的数学证明上，保证了基础模型推导的正确性和模型结果的精确度。

(2) 问题一 (1) 中的模型考虑了实际情况中角度信息存在误差的情况，该误差影响简单模型的求解结果，因此为避免角度测量不准带来的误差，模型利用了最小二乘法减小误差。

(3) 将题述的“略有偏差”进行定量分析，明确规定了略有偏差的无人机合适的偏移范围，以便模型中的准确性分析和精度分析。

(4) 问题一 (3) 和问题二采用迭代算法进行多个无人机的近似定位，并用计算机代码仿真模拟。

(5) 本文中建立的模型原创性高，文章中所有的模型都是自行推导建立的。

7.2 模型的缺点

(1) 该模型建立在偏差范围较小的前提下，因此该模型可在一定限度内得到精确的结果。但在无人机的位置出现较大的误差时，模型求解结果地精确度会出现下降，或者在迭代模型过程中可能会出现不收敛的现象。

(2) 问题一 (3) 和问题二用到的迭代模型仅根据计算机仿真实验确立，即本文章暂未给出迭代模型收敛的严格证明。

7.3 模型的推广

(1) 对于误差较大时，模型的精度下降以及可能出现的非收敛情况，可以通过对相关坐标进行动态调整加以解决，保证迭代算法的鲁棒性。

(2) 该模型仅适用于二维情形，但可以进一步推广到三维情形，以进一步解决实际问题。

(3) 本文建立的模型不只可以应用到无人机遂行编队飞行的纯方位无源定位中，还可以应用到其他各个科研领域，例如机器人设计、电子对抗以及雷达侦察等相关行业。

8 参考文献

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

[2] 王鼎, 张莉, 吴瑛.(2006). 基于角度信息的约束总体最小二乘无源定位算法. 中国科学 E 辑: 信息科学 (08),880-890.

9 附录

文件名	功能
polar2cart.m	将极坐标转化为笛卡尔坐标的 MATLAB 代码实现
calAG.m	计算两个向量之间的夹角的 MATLAB 代码实现
circle.m	根据两点及一个圆心角确定两个圆的圆心及半径的 MATLAB 代码实现
findCC.m	在 circle.m 函数中返回的两个圆中找到有效圆的 MATLAB 代码实现
calCDN.m	用最小二乘法计算多个圆的近似公共点的 MATLAB 代码实现
problem1a.m	问题一 (1) 模型的 MATLAB 代码实现
problem1c.m	问题一 (3) 模型的 MATLAB 代码实现
problem2.m	问题二模型的 MATLAB 代码实现
problem1a.py	问题一 (1) 的 Python 代码实现
problem1b.py	问题一 (2) 的 Python 代码实现

将极坐标转化为笛卡尔坐标的 MATLAB 代码实现：

```
function cart = polar2cart(polar)
    %{

    Convert a polar coordinate to its cartesian coordinate
        .

    ARGUMENT:
    polar:  a polar coordinate (r, theta).
            The angle theta should be in radians.

    RETURN:
    cart:   the converted cartesian coordinate (x, y).

    %}

    r = polar(1);
    theta = polar(2);

    cart = [0  0]';
```

```
19      cart(1) = r * cos(theta);
20      cart(2) = r * sin(theta);
21  end
```

计算两个向量之间的夹角的 MATLAB 代码实现

```
1  function alpha = calAG(x, y, z)
2      %{
3
4      Calculate the angle between the vectors y-x and z-x.
5
6      ARGUMENTS:
7      x, y, z:     three 2-dimensional vectors
8                   stand for the cartesian coordinates of
                       three points.
9
10     RETURNS:
11     alpha:  the angle between the vectors y-x and z-x in
           radians.
12
13     %}
14
15     a = y - x;
16     b = z - x;
17
18     alpha = acos(dot(a, b) / (norm(a) * norm(b)));
19 end
```

根据两点及一个圆心角确定两个圆的圆心及半径的 MATLAB 代码实现

```
1  function [R, c1, c2] = circle(a, b, alpha)
2      %{
3
4      Find two circles determined by two points and one
           circle angle.
```

```
5
6       ARGUMENTS:
7       a, b:    two 2-dimensional vectors
8                stand for two points on the determined circles
                     .
9       alpha:   the circle angle in radians.
10
11      RETURNS:
12      R:       the common radius of the two determined
            circles.
13      c1, c2:  two 2-dimensional vectors
14               stand for the centers of the two determined
                     circles.
15
16      %}
17
18      % The distance between a and b.
19      l = norm(a - b);
20
21      % The mid-point of a and b.
22      mid = (a + b) / 2;
23
24      % The unit perpenticular vector.
25      per = [b(2) - a(2); a(1) - b(1)] / l;
26
27      % By the Law of Sine.
28      R = l / (2 * sin(alpha));
29
30      % The common distance between mid and two centers. By
            Pythagorean Theorem.
31      d = sqrt(R^2 - (l / 2)^2);
32
33      % By geometry.
34      c1 = mid + d * per;
35      c2 = mid - d * per;
36  end
```

在 circle.m 函数中返回的两个圆中找到有效圆的 MATLAB 代码实现：

```
function c = findCC(R, c1, c2, x)
    %{

    Find the correct circle which has a smaller difference
        between R
    and the distance between its center and x.

    ARGUMENTS:
    R:      the common radius of two circles.
    c1, c2: two 2-dimensional vectors
            stand for the centers of two circles.
    x:      a 2-dimensional vector
            stands for the cartesian coordinate of a point
                .

    RETURNS:
    c:  the center of the correct circle.

    %}

    if abs(norm(c1 - x) - R) < abs(norm(c2 - x) - R)
        c = c1;
    else
        c = c2;
    end

end
```

用最小二乘法计算多个圆的近似公共点的 MATLAB 代码实现：

```
function cdn = calCDN(C, R)
    %{

```

```
4       Calculate the "intersection" of several circles by
            least squares.
5
6       ARGUMENTS:
7       C:  a 2 by n matrix
8           stores the cartsian coordinates of the centers of
                the circles.
9       R:  an n-dimensional vector stores the radiuses of the
            circles.
10
11      RETURNS:
12      cdn:    a 2-dimensional vector stands for the
            cartesian coordinate
13              of the "intersection".
14
15      %}
16
17      n = size(R, 1);
18
19      A = zeros(n, 2);
20
21      b = zeros(n, 1);
22
23      for i = 1:(n - 1)
24          A(i, :) = 2 * (C(:, i) - C(:, i + 1))';
25          b(i) = (sum(C(:, i).^2) - R(i)^2) - (sum(C(:, i +
                1).^2) - R(i + 1)^2);
26      end
27
28      A(n, :) = 2 * (C(:, n) - C(:, 1))';
29      b(n) = (sum(C(:, n).^2) - R(n)^2) - (sum(C(:, 1).^2) -
            R(1)^2);
30
31      cdn = lsqr(A, b);
32  end
```

问题一 (1) 模型的 MATLAB 代码实现：

```
real = [
    0 100 98 112 105 98 112 105 98 112;
    0 0 40.10 80.21 119.75 159.86 199.96 240.07 280.17
       320.28;
    ];

init = [
    0 100 100 100 100 100 100 100 100 100;
    0 0 40 80 120 160 200 240 280 320;
    ];

for i = 1:10
    real(:, i) = polar2cart([real(1, i), deg2rad(real(2, i
       ))]);
    init(:, i) = polar2cart([init(1, i), deg2rad(init(2, i
       ))]);
end

real

init

f0 = real(:, 1);

n = [5, 7, 10];

x = init(:, n(1));
y = init(:, n(2));
z = init(:, n(3));

rx = real(:, n(1));
ry = real(:, n(2));
rz = real(:, n(3));

alpha1 = calAG(rx, f0, ry);
alpha2 = calAG(rx, ry, rz);
```

```
alpha3 = calAG(rx, rz, f0);

C = zeros(2, 3);

R = zeros(3, 1);

[R(1), C1, C2] = circle(f0, ry, alpha1);
C(:, 1) = findCC(R(1), C1, C2, x);

[R(2), C1, C2] = circle(ry, rz, alpha2);
C(:, 2) = findCC(R(2), C1, C2, x);

[R(3), C1, C2] = circle(rz, f0, alpha3);
C(:, 3) = findCC(R(3), C1, C2, x);

calx = calCDN(C, R);

error = calx - rx
```

问题一 (3) 模型的 MATLAB 代码实现：

```
real = [
    0 100 98 112 105 98 112 105 98 112;
    0 0 40.10 80.21 119.75 159.86 199.96 240.07 280.17
       320.28;
    ];

init = [
    0 100 100 100 100 100 100 100 100 100;
    0 0 40 80 120 160 200 240 280 320;
    ];

for i = 1:10
    real(:, i) = polar2cart([real(1, i), deg2rad(real(2, i
       ))]);
    init(:, i) = polar2cart([init(1, i), deg2rad(init(2, i
```

```
        ))]);
end

cur = init;

last = zeros(2, 10);

C = zeros(2, 3);

R = zeros(3, 1);

tor = 1.0e-02;

cnt = 0;

while max(max(abs(cur - last))) > tor

    cnt = cnt + 1;

    last = cur;

    for i = 0:3

        a = 3 + i;
        b = 7 + i;

        for j = 1:2

            c = [cur(:, a) cur(:, 1) cur(:, 2) cur(:, b)];

            r = [real(:, a) real(:, 1) real(:, 2) real(:,
                b)];

            for k = 2:3

                for l = (k + 1):4

```

```
                        m = (3 - (4 - k) * (5 - k) / 2) + (l -
                            k);

                        alpha = calAG(r(:, 1), r(:, k), r(:, l
                            ));

                        [R(m), C1, C2] = circle(c(:, k), c(:,
                            l), alpha);

                        C(:, m) = findCC(R(m), C1, C2, c(:, 1)
                            );
                    end

                end

                cur(:, a) = calCDN(C, R);

                t = a;
                a = b;
                b = t;
            end

        end

    end

    error = cur - real

    cnt
```

问题二模型的 MATLAB 代码实现：

```
real = [
    0 50 152 184;
    0 0 60 46;
```

```
5       ];
6
7   init = [
8       0 50 150 180.28;
9       0 0 60 46.1;
10      ];
11
12  for i = 1:4
13      real(:, i) = polar2cart([real(1, i), deg2rad(real(2, i
            ))]);
14      init(:, i) = polar2cart([init(1, i), deg2rad(init(2, i
            ))]);
15  end
16
17  cur = init;
18
19  last = zeros(2, 4);
20
21  C = zeros(2, 3);
22
23  R = zeros(3, 1);
24
25  tor = 1.0e-03;
26
27  cnt = 0;
28
29  while max(max(abs(cur - last))) > tor
30
31      cnt = cnt + 1;
32
33      last = cur;
34
35      a = 3;
36
37      b = 4;
38
39      for j = 1:2
```

```
40
41          c = [cur(:, a) cur(:, 1) cur(:, 2) cur(:, b)];
42
43          r = [real(:, a) real(:, 1) real(:, 2) real(:, b)];
44
45          for k = 2:3
46
47              for l = (k + 1):4
48
49                  m = (3 - (4 - k) * (5 - k) / 2) + (l - k);
50
51                  alpha = calAG(r(:, 1), r(:, k), r(:, l));
52
53                  [R(m), C1, C2] = circle(c(:, k), c(:, l),
                        alpha);
54
55                  C(:, m) = findCC(R(m), C1, C2, c(:, 1));
56              end
57
58          end
59
60          cur(:, a) = calCDN(C, R);
61
62          t = a;
63          a = b;
64          b = t;
65      end
66
67  end
68
69  error = cur - real
70
71  cnt
72
73  real
74
75  init
```

```
76
77 cur
```

问题一 (1) 的 Python 代码实现：

```
1  import math
2  import random
3  from sympy import *
4  import numpy as np
5  from scipy.linalg import solve
6  from scipy.optimize import root,fsolve
7
8  # 全局变量
9  g_ix = 0
10 g_iy = 0
11 g_r = 0
12
13 # 得到理想极坐标和直角坐标
14 def getCoordinate(R):
15     polar_lst = []
16     vert_lst = []
17     for i in range(9):
18         a = i*40
19         polar_lst.append((R, a))
20         x = float(format(R*math.cos(math.radians(a))))
21         y = float(format(R*math.sin(math.radians(a))))
22         vert_lst.append((x, y))
23     polar_lst.insert(0,(0,0))
24     vert_lst.insert(0,(0,0))
25     for i in range(10):
26         print(polar_lst[i], vert_lst[i])
27     return polar_lst, vert_lst
28
29 ## 圆心坐标求解的辅助函数
30 def getCom(x):
31     global g_ix, g_iy, g_r
```

```
32      return np.array([
33              (x[0]-g_ix)**2 + (x[1]-g_iy)**2 -g_r**2,
34                      (x[0]**2 + x[1]**2 -g_r**2)])
35
36  # 根据两点和角度和得到圆心，根据待求坐标舍去无效圆
37  def calCir(x1, y1, x2, y2, a, xt, yt):
38      # print('-'*80)
39      l = ( (x1-x2)**2+(y1-y2)**2 )**(1/2)
40      r = l/2/math.sin(math.radians(a))
41      # r = l/math.sin(a)
42      # print('l: ',l)
43      # print('r: ',r)
44      d = (r**2-(l/2)**2)**(1/2)
45      x01 = (x1+x2)/2 + (d/l)*(y2-y1)
46      x02 = (x1+x2)/2 - (d/l)*(y2-y1)
47      y01 = (y1+y2)/2 + (d/l)*(x1-x2)
48      y02 = (y1+y2)/2 - (d/l)*(x1-x2)
49      coor_lst = [(x01,y01),(x02,y02)]
50      ans_lst = []
51      # print('求解过程: ')
52      for ix,iy in coor_lst:
53          # print('-'*80)
54          global g_ix
55          global g_iy
56          global g_r
57          g_ix = ix
58          g_iy = iy
59          g_r = r
60          result_root = root(getCom,[0,0])
61          result_fsolve = fsolve(getCom,[0,0])
62          # print(result_root)
63          # print("————————————————————————————————")
64          # print(result_fsolve)
65          ans_lst.append(tuple(result_fsolve))
66      # print('求解答案: ', ans_lst)
67      # dis1 = (ans_lst[0][0]-xt)**2+(ans_lst[0][1]-yt)**2
68      # dis2 = (ans_lst[1][0]-xt)**2+(ans_lst[1][1]-yt)**2
```

```
69      dis1 = (x01-xt)**2+(y01-yt)**2
70      dis2 = (x02-xt)**2+(y02-yt)**2
71      # print('距离: ',dis1, dis2)
72      ans = (x01,y01,r) if abs(dis1-r) < abs(dis2-r) else (
            x02,y02,r)
73      # print('最终结果',ans)
74      return ans
75
76  # 以三点得出角度
77  def getDegree(upoint,point_2,point_3):
78      a=math.sqrt((point_2[0]-point_3[0])*(point_2[0]-
            point_3[0])+(point_2[1]-point_3[1])*(point_2[1] -
            point_3[1]))
79      b=math.sqrt((upoint[0]-point_3[0])*(upoint[0]-point_3
            [0])+(upoint[1]-point_3[1])*(upoint[1] - point_3
            [1]))
80      c=math.sqrt((upoint[0]-point_2[0])*(upoint[0]-point_2
            [0])+(upoint[1]-point_2[1])*(upoint[1]-point_2[1]))
81      A=math.degrees(math.acos((a*a-b*b-c*c)/(-2*b*c)))
82      B=math.degrees(math.acos((b*b-a*a-c*c)/(-2*a*c)))
83      C=math.degrees(math.acos((c*c-a*a-b*b)/(-2*a*b)))
84      return A
85
86  # 对单个偏差无人机定位实际坐标
87  def findSingleFY(to_num_lst, from_num_lst):
88      ans = []
89      global vert_lst, R
90      from_couple_lst = [(from_num_lst[i],from_num_lst[i-1])
            for i in range(3)]
91      realx = vert_lst[to_num_lst[0]][0]+random.uniform
            (-15,15)
92      realy = vert_lst[to_num_lst[0]][1]+random.uniform
            (-15,15)
93      upoint = (realx,realy)
94      test_degree_lst = []
95      for i in range(3):
96          small_from_num_lst = from_couple_lst[i]
```

```
97            x1 = vert_lst[small_from_num_lst[0]][0]
98            y1 = vert_lst[small_from_num_lst[0]][1]
99            x2 = vert_lst[small_from_num_lst[1]][0]
100           y2 = vert_lst[small_from_num_lst[1]][1]
101           xt = vert_lst[to_num_lst[0]][0]
102           yt = vert_lst[to_num_lst[0]][1]
103           test_degree_lst.append(getDegree(upoint,(x1,y1),(
                  x2,y2)))
104           test_degree = test_degree_lst[i]
105           # print('test_degree:',test_degree)
106           ans.append(calCir(x1,y1,x2,y2,test_degree,xt,yt))
107       # print('-'*80)
108       # for i in range(3):
109           # print('final ans{0}'.format(i), ans[i])
110       # ans = [(x,y,r),*3]
111       whole_lst = []
112       for i in range(3):
113           x1 = ans[i][0]
114           x2 = ans[i-1][0]
115           y1 = ans[i][1]
116           y2 = ans[i-1][1]
117           r1 = ans[i][2]
118           r2 = ans[i-1][2]
119           # print(x1,y1,r1)
120           # print(x2,y2,r2)
121           lst = []
122           lst.append(2*(x1-x2))
123           lst.append(2*(y1-y2))
124           lst.append((x1**2+y1**2-r1**2)-(x2**2+y2**2-r2**2)
                  )
125           whole_lst.append(lst)
126       A = np.matrix([whole_lst[0][0:2],whole_lst[1][0:2],
              whole_lst[2][0:2]])
127       B = np.matrix([[whole_lst[0][2]],[whole_lst[1][2]],[
              whole_lst[2][2]]])
128
129       result = solve(A.T*A, A.T*B)
```

```
        result = (float(result[0]),float(result[1]))
        print('-'*80)
        print('发送飞机编号',from_num_lst)
        print('发送飞机组合',from_couple_lst)
        print('接收飞机编号',to_num_lst)
        print('标准坐标',vert_lst[to_num_lst[0]])
        print('实际坐标',(realx,realy))
        print('接收角度',test_degree_lst)
        print('计算坐标',result)

if __name__ == '__main__':
    R = 100
    polar_lst, vert_lst = getCoordinate(R)
    for i in range(100):
        # 生成随机数
        to_num = random.randint(1,9)
        plane1 = random.randint(1,9)
        plane2 = random.randint(1,9)
        # to_num = 1
        # plane1 = 7
        # plane2 = 4
        if plane1 == plane2 or plane1 == to_num or plane2
            == to_num:
            continue
        findSingleFY([to_num],[0,plane1,plane2])
```

问题一 (2) 的 Python 代码实现：

```
import math
import random
from sympy import *
import numpy as np
from scipy.linalg import solve
from scipy.optimize import root,fsolve

g_ix = 0
g_iy = 0
```

```
g_r = 0

# 得到标准极坐标和直角坐标
def getCoordinate(R):
    polar_lst = []
    vert_lst = []
    for i in range(9):
        a = i*40
        polar_lst.append((R, a))
        x = float(format(R*math.cos(math.radians(a))))
        y = float(format(R*math.sin(math.radians(a))))
        vert_lst.append((x, y))
    polar_lst.insert(0,(0,0))
    vert_lst.insert(0,(0,0))
    for i in range(10):
        print(polar_lst[i], vert_lst[i])
    return polar_lst, vert_lst

# 以三点得出角度
def getDegree(upoint, point_2, point_3):
    a=math.sqrt((point_2[0]-point_3[0])*(point_2[0]-
        point_3[0])+(point_2[1]-point_3[1])*(point_2[1] -
        point_3[1]))
    b=math.sqrt((upoint[0]-point_3[0])*(upoint[0]-point_3
        [0])+(upoint[1]-point_3[1])*(upoint[1] - point_3
        [1]))
    c=math.sqrt((upoint[0]-point_2[0])*(upoint[0]-point_2
        [0])+(upoint[1]-point_2[1])*(upoint[1]-point_2[1]))
    A=math.degrees(math.acos((a*a-b*b-c*c)/(-2*b*c)))
    B=math.degrees(math.acos((b*b-a*a-c*c)/(-2*a*c)))
    C=math.degrees(math.acos((c*c-a*a-b*b)/(-2*a*b)))
    return float(format(A,'.2f'))

# 在已知自身编号和标准坐标且已知0和1两个无人机的标准坐标情
    况下，得知其他编号的无人机参与发射信号后可以得知的夹角
    坐标
def getAllLst(pointu, point0, point1, to_num):
```

```
40      lst1 = []
41      for i in range(10):
42          lst0 = []
43          if i != 0 and i!= 1 and i!= to_num:
44              lst0.append(i)
45              point2 = vert_lst[i]
46              lst0.append(getDegree(pointu, point0, point2))
47              lst0.append(getDegree(pointu, point1, point2))
48              lst1.append(lst0)
49      return lst1
50
51  # 极坐标转化为直角坐标
52  def polarToVert(R, a):
53      x = R*math.cos(math.radians(a))
54      y = R*math.sin(math.radians(a))
55      return (x,y)
56
57  if __name__ == '__main__':
58      R = 100
59      polar_lst, vert_lst = getCoordinate(R)
60      count = 0
61      time = 100000
62      delta_dis = 13
63      delta_degree = 2
64      for m in range(time):
65          to_num = random.randint(2,9)
66          pointu = vert_lst[to_num]
67          point0 = vert_lst[0]
68          point1 = vert_lst[1]
69          allPosLst = getAllLst(pointu,point0,point1,to_num)
70          # for i in range(len(allPosLst)):
71          #     print(allPosLst[i])
72          realR = polar_lst[to_num][0]+random.uniform(-
                delta_dis,delta_dis)
73          reala = polar_lst[to_num][1]+random.uniform(-
                delta_degree,delta_degree)
74          realx, realy = polarToVert(realR, reala)
```

```
75          while true:
76              plane2 = random.randint(2,9)
77              if plane2 != to_num:
78                  break
79          point2 = vert_lst[plane2]
80          pointreal = (realx, realy)
81          degree1 = getDegree(pointreal,point0,point2)
82          degree2 = getDegree(pointreal,point1,point2)
83          curnum = -1
84          curdis = 10**9
85          for i in range(7):
86              x1 = allPosLst[i][1]
87              y1 = allPosLst[i][2]
88              x2 = degree1
89              y2 = degree2
90              dis = (x1-x2)**4+(y1-y2)**2
91              if dis < curdis:
92                  curdis = dis
93                  curnum = i
94          print('接收：',to_num,end ='\t')
95          print(degree1,degree2, end='\t')
96          print('坐标：',format(realx,'.2f') ,format(realy,'
               .2f'),end='\t')
97          print('待确定：',plane2, end= '\t')
98          print('确定编号为：',allPosLst[curnum][0],end = '\
               t')
99          if allPosLst[curnum][0] == plane2:
100             count += 1
101             print('true')
102         else:
103             print('false')
104     print('{0}次准确率为：{1}%'.format(time,count*100/time
            ))
```