基于角度定位法的无人机纯方位无源定位

摘要

纯方位无源定位法可以很好地在避免外界干扰的情况下调整无人机编队的队形。本文所研究的无人机编队有固定的编号及固定的相对位置且规定无人机所接受到的方向信息为其与任意两架发射信号无人机连线之夹角。本文通过几何分析、代数运算与仿真模拟，建立并验证了无人机队列在多种情况下的纯方位无源定位法。

针对问题一的第（1）问，首先将复杂的排列情况通过以处于优弧还是处于劣弧为准则，分成两大类情况，进行**分类讨论**，并且给出通解。接着以所在的弧为准则，进行**分类讨论**。在讨论中，借助使用**角度定位法，即通过其他无人机发送的角度信息，引进假设的半径长度，可以确定无人机的轨迹圆，使无人机得到一个自身的确切位置，并最终通过极坐标确定自己的行进方向与距离，从而抵达自己的正确位置。**

针对问题一的第（2）问，采用**枚举**法。首先通过枚举，得出了至少增添一架无人机情况是不可行的。接着，通过对**枚举**两架无人机的所有情况，并结合角度定位法，在算法中得出了无人机的具体位置，验证了增添两架无人机的情况是可行的。最后得出结论，至少需要两架无人机发射信号才能精确定位。

针对问题一的第（3）问，首先选取 FY00 与 FY01，并从余下的无人机抽取两架作为发射信号的无人机。首先假设所有发射信号的无人机都是无偏差的。接着通过三点共圆，再根据已知数据，构建 **Taylor 定位算法**并结合**最小二乘法**，得到所调整无人机的一个“伪”坐标。然后，无人机可以借助伪坐标与正确坐标，多次进行**迭代**调整，不断减小误差，最终逐渐收敛至其正确位点。最后，仿真模拟实验验证了我们模型的正确性。

针对问题二，首先补充假设，然后对所有无人机进行分类。仍然采用与问题一第（3）问类似的寻找伪坐标-**迭代**方法。在一轮调整中，首先选取中心三架无人机发射信号，接着再选取边角三架无人机发射信号。通过**反复迭代**，最终使所有无人机收敛于其正确位点。此外，对于存在特殊情况的无人机，**分类讨论**处理方法。最后，仍然通过仿真模拟实验验证了我们的模型。

最后，给出每个模型的优缺点及评价。

关键词：角度定位法、最小二乘法、Taylor 定位算法、纯方位无源定位

一：问题重述

1.1 问题背景

在互联网技术发展蓬勃的 21 世纪，基于互联网技术的物联网、人工智能、云计算等为核心的技术变革将进入高速拓展期，并对大到世界格局，小到人民群众的购物等产生前所未有的影响。无人机技术便是这类先进生产力的重要代表之一。早在 2003 年，我国就有相关的政策对该行业进行规范、约束。并且，自 2017 年以来，我国对无人机技术的管理、支持力度越来越大，更在 2019、2020 年密集出台多个政策与标准，以推动该行业健康发展。国内更是出现了像大疆这样富有活力与竞争力的无人机巨头。无人机除去单独执行任务外，它还需要具有在一些复杂的环境如沙漠、高原，甚至战场完成编队飞行任务的能力。在这种情况下，传统的无人机定位方式（如定位卫星导航）并不能很好地胜任这类任务。

在执行遂行编队飞行任务时，无人机为了减少外界对飞行的干扰，应尽量减少对外发射电磁波的频率。采用纯方位无源定位的方法来调整无人机编队的队形可以很好地满足上述需求。纯方位无源定位是指借助编队中几架无人机发出的信号，其余无人机接受其信号，提取方位信息，并调整自身位置，从而保持编队队形的低信息依赖的定位方法。

1.2 问题提出

根据上述背景，本团队需要解决以下问题：

1. 在纯方位无源定位的情况下，10 架无人机组成圆形编队，且其中编号 FY00 的无人机处在圆心，余下 9 架无人机（编号为 FY01~FY09）均匀分布在圆周上。
   - 1.1. 在 FY00 与另 2 架已知编号的无人机发射信号且其位置无偏差的时，建立其余略有偏差被动接受信号无人机的定位模型。
   - 1.2. 某一略有偏差的无人机在接收到无偏差的编号 FY00 和 FY01 的无人机发射的信号的前提下，另外接收到编队中若干同样无偏差但未知编号的无人机发射的信号。除 FY00 和 FY01 外，还需要几架无人机，才能使该无人机精准定位？
   - 1.3. 根据表 1 所给的无人机初始位置，每次选择编号为 FY00 的无人机与圆周上最多 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来调整自身位置。给出其余无人机多次具体调整的方案。
2. 同样在纯方位无源定位的情况下，设计如图 3 的锥形编队的无人机位置调整方案。

其中，表一给出了在初始状况，十架无人机的极坐标方位，包括了各个无人机的编号、极坐标的极角、极径；图 3 给出了锥形编队的具体编队的形式。

二：问题分析

2.1 问题 1 第（1）问的分析

问题 1 的第（1）问要求我们在处于编队中心的无人机 FY00 与编队中另两架无人机发射信号且编队其他略有偏差的无人机接收信号的时候，建立后者的定位模型。本题给定了发射信号的无人机的编号，但没有给出具体的假设。我们不妨

设发射信号的非 FY00 的两架无人机分别为 FY0i 与 FY0j，且 i>j。与此同时，取决于接受信号的无人机的位置，我们又可以将无人机的调整方案分为处于劣弧类与处于优弧类。再根据各个无人机所处位置，我们可以借助角度定位法，来寻找其位置，并最终在极坐标中给出具体的调整方案。

2.2 问题 1 第（2）问的分析

问题 1 的第（2）问要求我们在某一略有偏差的无人机在接收到无偏差的编号 FY00 和 FY01 的无人机发射的信号的前提下，另外接收到编队中若干同样无偏差但未知编号的无人机发射的信号。最后求出除 FY00 和 FY01 外，还需要几架无人机，才能使该无人机精准定位。

我们拟采用枚举法，即假定发射信号的无人机的具体编号，并遍历其余所有无人机，然后通过算法证明其是否能满足精确定位的要求。

首先，我们先证明仅需要一架的情况不成立。我们假定目前调整位置的无人机为 FY0M，发射信号的无人机的编号为 FY0i。由于我们只需要证明该情况不成立，故举出一个反例即可。我们假定 M=8，并遍历 FY0i，通过角度定位法，将计算出七个位置，而其中有六个是噪音，无人机无法精确地判断自己的位置。故一架的情况不成立。

然后，我们需要证明还需要两架的情况成立。同样如上，我们再假设新加入发射信号的无人机编号为 FY0j。此时我们通过时间复杂度为n^2的遍历，即在固定 FY0i 的情况下，遍历 FY0j，并在每次遍历完成后令 i 加一，重复遍历的过程。最终计算出三个三点共圆做出的圆的交点，得到了 FY0M 的准确位置。

2.3 问题 1 第（3）问的分析

问题 1 的第（3）问要求我们根据表 1 所给的无人机初始位置，每次选择编号为 FY00 的无人机与圆周上最多 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来调整自身位置并给出其余无人机多次具体调整的方案。

在第一轮的无人机选取中，我们不妨选定 FY00 与 FY01 并自余下的无人机中两两选择距离最远的无人机作为发射信号的无人机。我们将这两架无人机称为 FY0i 与 FY0j，其中 j=i+4，i=1,2,3,4。此时，我们假设 FY00 和 FY01 位置无偏差，其余无人机横纵坐标的误差服从均值为零的正态分布。但在算法中，我们假设发射信号的无人机位置都是无偏差的。实际上，由于无人机的偏差程度相较于其距离而言极小且服从正态分布，我们错误的假设对调整方案得出的最终结果平均来看几乎没有影响。我们令目前我们选定做调整的无人机为 FY0M，选定 FY0M 与 FY00，并选择 FY01、FY0i、FY0j 分别确定轨迹圆。由三点共圆，且弦长和圆心角已知，故上述操作是可实现的。此时，我们可据上述数据，使用最小二乘法与构建 TayIor 级数展开定位模型，得到 FY0M 的一个“伪”坐标。而我们通过对极坐标的转换，是可以知道无人机的目标位点的。无人机可以借助伪坐标与正确坐标，多次进行迭代调整，最终逐渐收敛至自己的正确位点。最后，我们进行了仿真模拟，验证了模型的可行性。

2.4 问题二的分析

问题二要求我们同样在纯方位无源定位的情况下，设计锥形编队的无人机位置调整方案。相比于第一题，问题二缺乏一些关键的信息。因此在做第二问之前，除去与第一问共用的基本假设，我们还需要自行做出额外的假设。除此

之外，问题二采取的是锥形编队，因此对我们所设计的方法而言，其需要做出一定的分类，才能使所有的无人机抵达正确的位点。

本题我们假设 FY01 和 FY15 位置无偏差，其余无人机 x,y 的偏移量均服从均值为零的正态分布。这也为之后的仿真模拟提供了条件。在算法中，我们也假设发射信号的无人机位置都是无偏差的。此外，我们以 FY13 为原点，FY13 指向 FY01 作为 X 轴正方向，FY13 指向 FY11 作为 Y 轴正方向，构建了直角坐标系。

我们的调整方案一轮分为两步。第一步，我们选中 FY05、FY08、FY09 作为发出信号的无人机。对于所有无人机而言，如 FY02 这类与中心三架无人机可能存在共线的情况的无人机，我们将分类讨论。与第一题第（3）问类似地，做三点共圆，并据上述数据，使用最小二乘法与构建 TayIor 级数展开定位模型，得到受信号无人机的一个“伪”坐标。且由于我们构建了直角坐标系。无人机可以借助“伪”坐标与正确坐标，前往距离正确目标更近的中间目标。第二步，我们选中 FY01、FY11、FY15 三架无人机发射信号，同样寻找到一个正确的目标。重复上述过程，多次进行迭代调整，无人机最终将逐渐收敛至自己的正确位点。最后，我们同样进行了仿真模拟，验证了模型的可行性。

三：基本假设

1. 假设无人机严格按照题设排序，包括相对位置不变以及第一题编号为 FY00 的无人机位于圆心。
2. 假设无人机能够精确地按某确定的角度飞行一定的距离。
3. 假设无人机所发射与接收的信号中的方向信息与编号信息真实可靠。
4. 假设无人机均在同一高度飞行。
5. 忽略外界因素如地球的曲率、空气中存在的电磁波对无人机各项功能运作的影响。

四：符号说明

符号	符号说明	单位
M	选取无人机的编号	无
α_k	被动无人机接受的角度信息	π

注：重复的符号及未出现符号以出现处为准

五：问题 1 第（1）问的模型建立与求解

5.1 问题的分析

问题 1 的第（1）问要求我们在处于编队中心的无人机 FY00 与编队中另两架无人机发射信号且编队其他略有偏差的无人机接收信号的时候，建立后者的定位模型。本题给定了发射信号的无人机的编号，但没有指定具体的发射信号的无人机。

我们不妨设发射信号的非 FY00 的两架无人机分别为 FY0i 与 FY0j，且 i>j。首先，取决于接受信号的无人机的位置，我们可以将无人机的调整方案分为处于劣弧类与处于优弧类。通过对 i、j 情况的探讨，我们可以将复杂的实际情况，简化成处于劣弧类的无人机调整方案与处于优弧类的无人机调整方案。其次，根据

各个无人机所处位置，我们可以借助三角形的边角关系并使用正弦定理，由此，我们可以得到无人机确切的位置，并最终在极坐标中给出具体的调整方案。

5.2 模型的建立

由于本题主要使用几何学的方法进行求解，无人机处于优弧与劣弧的情况便不得不分开进行处理。我们不妨假定我们所调整的无人机的编号为 FY0M，其中 $M \in [1,9]$ 且 $M \neq$ i,j。

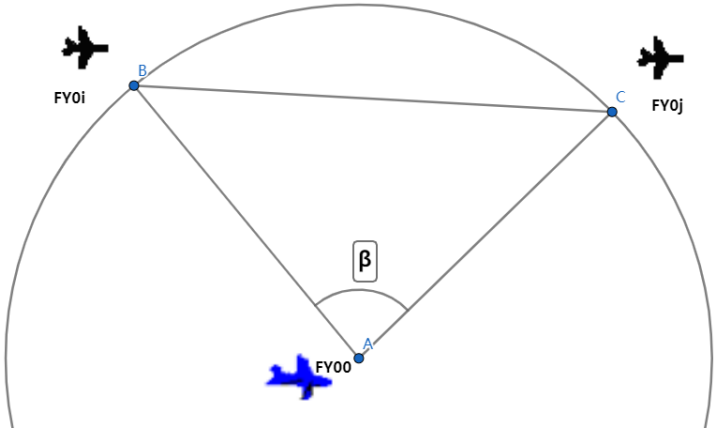


图 5.1 优劣弧情况探讨

此处我们还需要讨论 i 与 j 之间的差，以确认优弧的位置。如图 5.1 所示，令 FY00 的位置为 A，FY0i 的位置为 B，FY0j 的位置为 C，令 $\angle CAB$ 的度数为 β，且严格规定 β 的度数为自 AC 逆时针旋转至 AB 的度数。当 $\beta < \pi$ 时，即：

$$(i-j)*\frac{2}{9}\pi < \pi$$

有 $i-j<5$ 作为其分类标准。由此我们可知：

$$\begin{cases} \text{优弧情况：} i-j<5 \text{ 且 } M>i \text{ 或 } M<j;\ i-j>5 \text{ 且 } i<M<j \\ \text{劣弧情况：} i-j>5 \text{ 且 } M>i \text{ 或 } M<j;\ i-j<5 \text{ 且 } i<M<j \end{cases}$$

这样我们就能将复杂的无人机排列情况二分为优弧情况与劣弧情况。以下我们分情况建立无人机的定位模型。

5.2.1. 优弧情况模型的建立与求解

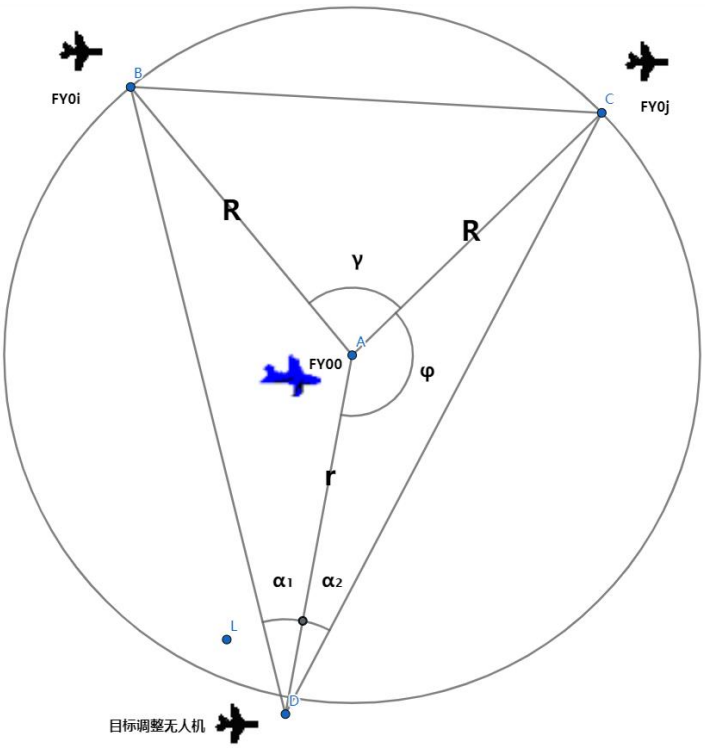


图 5.2：优弧情况

如图 5.2 所示，建立模型。令 FY00 的位置为 A，FY0i 的位置为 B，FY0j 的位置为 C，目标调整无人机的位置为 D。令 $\angle CAB$ 的度数为 γ，$\angle CAD$ 的度数为 φ；设圆形编队所在的圆的半径为 R，令 AD 的长度为 r（r≠R）。依题意，设 $\angle ADB$ 的度数为 α_1，$\angle ADC$ 的的度数为 α_2，且不难知道 $\alpha_3=\alpha_1+\alpha_2$。

以下为模型的求解过程。

由题目，我们可知，各个无人机是均匀分布的，且我们知道发出信号的 FY0i 与 FY0j 的编号，并且其在正确的位置上。故我们知道

$$\gamma=\frac{2(i-j)}{9}\pi \quad \cdots\cdots ①$$

以 FY00 所在位置为极点，由 FY00 指向 FY0j 的射线为极轴，建立极坐标系，以逆时针为正方向。则各点坐标为：A（0,0）,C（R,0）,B（R, γ）,D(R,- γ)。

在 ΔADC 中，由正弦定理得：

$$\frac{R}{\sin\alpha_2}=\frac{r}{\sin(\pi-\alpha_2-\varphi)}=\frac{r}{\sin(\alpha_2+\varphi)}\cdots\cdots ②$$

在 ΔABD 中，由正弦定理得：

$$\frac{R}{\sin\alpha_1}=\frac{r}{\sin(\pi-\alpha_1-(2\pi-\gamma-\varphi))}=\frac{r}{\sin(\alpha_1-\gamma-\varphi)}\cdots\cdots ③$$

由②、③作商，得：

$$\frac{\sin\alpha_2}{\sin\alpha_1}=\frac{\sin(\pi-\alpha_2-\varphi)}{\sin(\pi-\alpha_1-(2\pi-\gamma-\varphi))}=\frac{\sin(\alpha_2+\varphi)}{\sin(\alpha_1-\gamma-\varphi)}\cdots\cdots④$$

对④进行变换，有：

$$\frac{\sin\alpha_1}{\sin\alpha_2}=\frac{\sin(\alpha_1-\gamma)-\cos(\alpha_1-\gamma)tan\varphi}{\sin\alpha_2+cos\alpha_2 tan\varphi}\cdots\cdots⑤$$

对⑤进行求解，有：

$$tan\varphi=\frac{sin\alpha_2\sin(\alpha_1-\gamma)-\sin\alpha_1 sin\alpha_2}{\sin\alpha_1 cos\alpha_2+sin\alpha_2\cos(\alpha_1-\gamma)}\cdots\cdots⑥$$

代入①对其进行求解并求取φ的值，有：

$$\varphi=\arctan\frac{sin\alpha_2\sin(\alpha_1-\gamma)-\sin\alpha_1 sin\alpha_2}{\sin\alpha_1 cos\alpha_2+sin\alpha_2\cos(\alpha_1-\gamma)}\cdots\cdots⑦$$

与此同时，由②，我们有

$$r=\frac{R\sin(\alpha_2+\varphi)}{\sin\alpha_2}\cdots\cdots⑧$$

此时无人机即可根据上式得到自己的精确位置（r，π-φ），通过定向移动校正自身位置，并到达其正确位点（R，$(M-j)*\frac{2}{9}\pi$）。

5.2.2. 劣弧情况模型的建立与求解

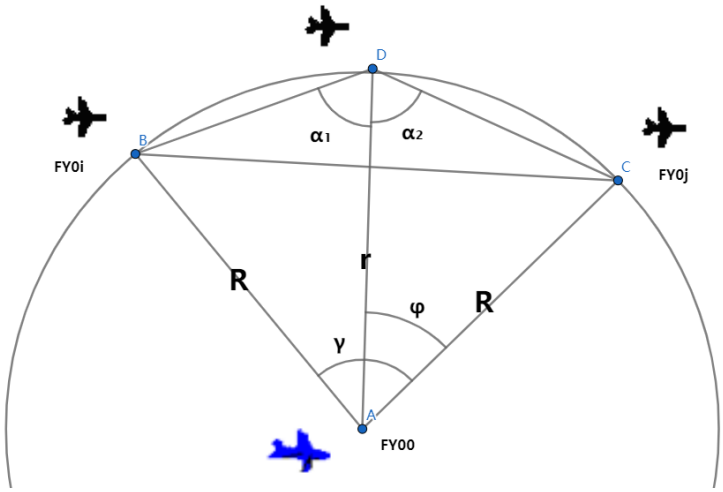


图 5.3：劣弧情况

如图 5.3 所示，建立模型。令 FY00 的位置为 A，FY0i 的位置为 B，FY0j 的位置为 C，目标调整无人机的位置为 D。令$\angle CAB$的度数为γ，$\angle CAD$的度数为φ；设圆形编队所在的圆的半径为 R，令 AD 的长度为 r（r≠R）。依题意，设$\angle ADB$的度数为α_1，$\angle ADC$的的度数为α_2，且不难知道$\alpha_3=\alpha_1+\alpha_2$。

根据我们的推算，劣弧情况模型的求解与优弧情况基本一致，而文章篇幅

有限，故此处不再过多赘述。我们同样可以得到下式：

$$r = \frac{R\sin(\alpha_2+\varphi)}{\sin\alpha_2} \cdots\cdots ⑨$$

此时无人机同样可根据上式得到自己的精确位置（r，π-φ），通过定向移动校正自身位置，并到达其正确位点（R，$(M-j)*\frac{2}{9}\pi$）。

六：问题一第（2）问的模型建立与求解……

6.1 问题分析

问题 1 的第（2）问要求我们在某一略有偏差的无人机在接收到无偏差的编号 FY00 和 FY01 的无人机发射的信号的前提下，另外接收到编队中若干同样无偏差但未知编号的无人机发射的信号。最后求出除 FY00 和 FY01 外，最少还需要几架无人机，才能使该无人机精准定位。

我们拟采用枚举法，即假定发射信号的无人机的具体编号，并遍历其余所有无人机，然后通过算法证明其是否能满足精确定位的要求。

由于题目需要求出最少需要的无人机数，那么我们不妨从需要一架的情况开始。因此，我们先证明一架的情况不成立。我们假定目前调整位置的无人机为 FY0M，发射信号的无人机的编号为 FY0i。由于我们只需要证明该情况不成立，故举出一个反例即可。我们假定 M=8，并令 FY0i 遍历其所有可行值。由于其情况与第一问完全相同，故我们可以借助第一题的方法，即角度定位法，计算出七个可能的位置，而其中有六个是不易排除的噪音，因此无人机无法精确地判断自己的位置。故一架的情况不成立。

然后，根据我们的后验知识，我们发现两架的情况能满足精确定位的需求。不过，我们仍然需要证明其成立。除保持假设同样如上，我们再假设新加入发射信号的无人机编号为 FY0j，且规定 i<j。此时我们通过时间复杂度为n^2的遍历，即在固定 FY0i 的情况下，令 FY0j 遍历其所有可行值，并在每次遍历完成后令 i 加一，反复遍历。最终我们将得到有且仅有的三圆共点，其即为 FY0M 的准确位置。

6.2 模型的构建与求解

6.2.1 证明仅添加一架无人机的情况是不可行的

首先，我们需要证明仅添加一架无人机的情况是不成立的。我们首先假定目前需要校正的无人机是 FY08，向程序中输入其编号与我们生成的角度信息。具体程序的结构我们借助伪代码来阐述。我们通过假设发射信号的无人机，可以得到 7 个可能的位置。并且，在这 7 个可能的位置中，存在着难以分开的噪音。因此，如果盲目地去排除噪音，我们很容易将正确的位置当成错误的值排除，从而使我们的结果产生偏差。

Algorithm 1 一架无人机的枚举算法

Input： 所求取无人机的编号 M，无人机收到的角度信息 α_1、α_2

Output： M 所有的可行位置 A_i

function OPTIMIZE（M）
$i \leftarrow$ **2**
while $i \leq 9$ **do**
while $i = M$ **do**
skip
$A_i \leftarrow$ **get** M **'s position**
return A_i=possible position of M
$i \leftarrow i+1$
end while
end function

上述伪代码需要着重阐述的部分是如何获取 M 的位置。由于我们假定了发出信号的无人机的编号，那么我们就获取了和第(1)问等量的信息。因此，我们可以将角度定位法迁移至本题。此时我们借用第(1)问所得的公式⑦与⑨，就可以知道在某假设情况下，无人机的精确位置为（r，π-φ）。程序的预备代码参见附录 1。上述算法的详细的代码请参见附录 2。

6.2.2 证明添加两架无人机的情况是可行的

6.2.2.1 圆方程的推导

如图 6.1 所示，令 A 的坐标为(x_1, y_1),B 的坐标为(x_2, y_2)，C 的坐标为(x_3, y_3)，令圆心的坐标为(x_r, y_r)，圆的半径为 R，$\angle ACB$ 的角度为α。

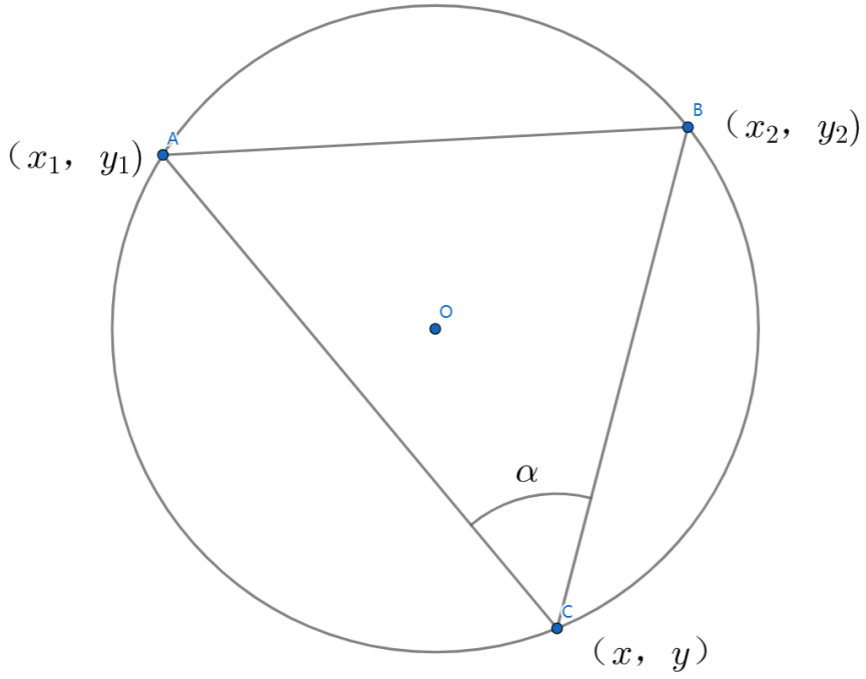


图 6.1：圆方程的推导

由向量的性质，有下式：

$$\left|\overrightarrow{CA}\bullet\overrightarrow{CB}\right|=|CA|\bullet|CB||\cos\alpha|$$

及

$$\left|\overrightarrow{CA}\times\overrightarrow{CB}\right|=|CA|\bullet|CB|\sin\alpha$$

我们借助中间变量$|CA|\bullet|CB|$，不难有下式：

$$|CA|\bullet|CB|=\left|\frac{\overrightarrow{CA}\bullet\overrightarrow{CB}}{\cos\alpha}\right|=\left|\frac{\overrightarrow{CA}\times\overrightarrow{CB}}{\sin\alpha}\right|\ldots\ldots⑨$$

当$\alpha\in(0,\pi/2)$，代入向量的表示式，可得

$$\frac{x^2+y^2-(x_1+x_2)x-(y_1+y_2)y+x_1x_2+y_1y_2}{\cos\alpha}=\frac{x_1y_2-x_2y_1+(y_1-y_2)x+(x_2-x_1)y}{\sin\alpha}$$

将其展开，有：

$$x^2+y^2-(x_1+x_2+\frac{y_2-y_1}{\tan\alpha})x-(y_1+y_2+\frac{x_2-x_1}{\tan\alpha})y=\frac{x_1y_2-x_2y_1}{\tan\alpha}-(x_1x_2+y_1y_2)$$

$$\ldots\ldots⑩$$

$\alpha\in(\pi/2,\pi)$的情况同理。

联立圆的方程，我们不难有以下公式：

$$x_r=\frac{1}{2}\left[x_1+x_2\pm\frac{1}{\tan\alpha}(y_1-y_2)\right]\ldots\ldots⑪$$

$$y_r=\frac{1}{2}\left[y_1+y_2\mp\frac{1}{\tan\alpha}(x_1-x_2)\right]\ldots\ldots⑫$$

$$(x-x_r)^2+(y-y_r)^2=R^2\ldots\ldots⑬$$

$$R^2=\frac{(x_1-x_2)^2+(y_1-y_2)^2}{4\sin^2\alpha}\ldots\ldots⑭$$

6.2.2.2 借助模型证明结论

与 6.2.1 方法类似，在本结论的证明中，我们仍然使用枚举法。在固定 FY0i 的情况下，令 FY0j 遍历其所有可行值，并在每次遍历完成后令 i 加一，接着重复 FY0j 的遍历过程。在遍历的过程中，我们会得到有且仅有的一个存在的坐标，其即为 FY0M 的坐标。具体的程序请参看附录 3。此处我们用伪代码来表示程序的构造。

Algorithm 2 两架无人机的枚举算法

Input：所求取无人机的编号 M，无人机收到的角度信息 α_1、α_2、α_3

Output：M 的可行位置 A

function OPTIMIZE（M）

```
i ← 2
j ← 2
while i ≤ 9 do
    while i = M do
        skip
    while j ≤ 9 do
        while j = M || j = i do
        skip
        A ← get M 's position
        j ← j+1
    end while
    i ← i+1
    j ← i
end while
    return A=position of M
end function
```

本结论的证明中使用的求坐标方法与 6.2.1 有较大的差异。借助推导的公式⑪、⑫、⑬、⑭，我们可以构建出如图 6.2 的三个圆，并且由于我们是遍历所有 i、j 的情况，所以其坐标是可知的。将 FY0i、FY0j、FY01 的坐标与α_1、α_2、α_3投入计算，我们可以得到图 6.2 中三个圆的具体方程，且可以求出它们的交点。根据我们的后验经验，FY0M 的坐标有且仅有一个点，而且它是图 6.2 中三个圆的具体交点。只有当我们遍历至正确情况时，三个圆有共同交点，且其为 FY0M 的具体坐标。由是我们得到了 FY0M 的正确坐标，即得添加两架无人机的情况是可行的。

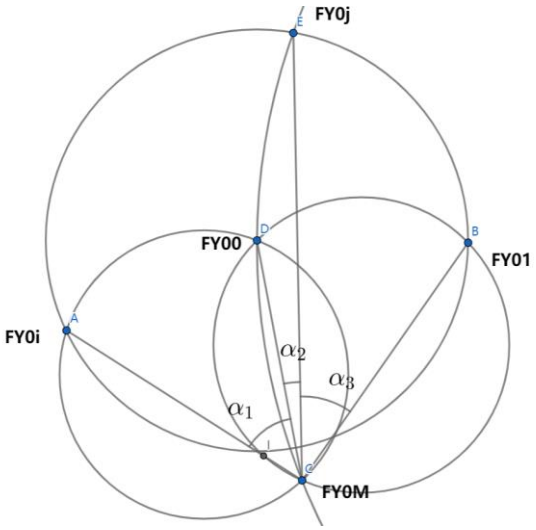


图 6.2：三点共圆的情况

6.3 结论

由 6.2.1，仅添加一架无人机的情况是不可行的，而再由 6.2.2，添加两架无人机的情况是可行的，我们可以给出结论，即除了 FY00 与 FY01 之外，至少需要两架无人机，才能够有效地实现无人机的无源定位。

七：问题一第（3）问的模型建立与求解……

7.1 问题的分析

问题 1 的第（3）问要求我们根据表 1 所给的无人机初始位置，每次选择编号为 FY00 的无人机与圆周上最多 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来调整自身位置并给出其余无人机多次具体调整的方案。

在第一轮的无人机选取中，我们不妨选定 FY00 与 FY01 并两两选取距离最远的飞机，如选定 FY02 则同时选定 FY06。由后验经验，我们发现选取距离最远的飞机，最终在代码的拟合效果是最好的。我们将这两架无人机称为 FY0i 与 FY0(i+4)。我们令目前我们选定做调整的无人机为 FY0M。

此时，在算法中，我们假设所有发射信号的无人机都是无偏差的。然而，由题所述，无人机的位置偏差是极小的。根据对表 1 数据的分析，我们发现无人机最大的位置偏差与无人机圆形队列的半径相比，最大不超过 12.01%。由此，我们即可借助这个假设极大地简化我们的问题，并且，据我们最终代码的推算结果，我们“错误”的假设对调整方案得出的最终结果几乎没有影响。

首先，选定 FY0M 与 FY00 两点，并选择 FY01、FY0i、FY0j 分别做圆。由三点共圆，上述操作是可实现的，且由于所有发射信号的无人机都是无偏差的，其间距离也是可知的。此时，我们可据上述数据，并且使用最小二乘法，构建 Taylor 级数展开定位模型，从而得到编号为 FY0M 的无人机的一个“伪”坐标，并将其表示于直角坐标系。通过对极坐标的转换，无人机所应调整至的正确位点是可知的。由于无人机接受到编号为 FY00 与 FY01 的无人机的信号，故我们认定其可以借助伪坐标与正确坐标，到达一个可能比目前位置更接近目标位置的中间点。在无人机横纵坐标原始误差服从均值为零的正态分布且方差较小的情况下，无人机可多次进行迭代调整，并最终逐渐收敛至自己的正确位点。

具体的迭代流程图如下：

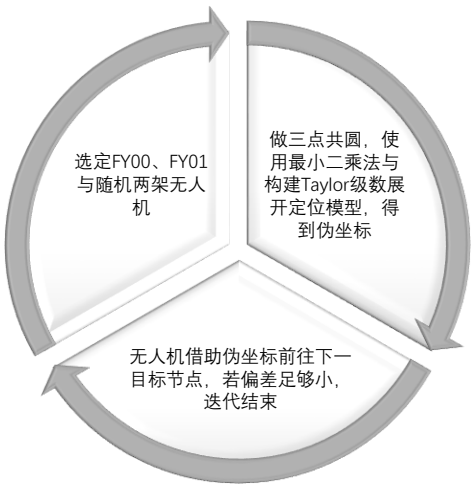


图 7.1：迭代流程图

7.2 模型的建立

首先我们可以选定无人机 FY00 与 FY01 发射信号。以 FY00 为原点，FY00 指向 FY01 为 X 轴的正方向，建立直角坐标系。我们假设所有发射信号的无人机都是没有偏差的，以近似处理。

7.2.1 数据预处理

相较于极坐标，直角坐标系能够提供更为直观的观感。对于无人机的移动而言，直角坐标系在计算中的便利性远胜过极坐标。因此，在建立模型之前，我们首先借助了 Matlab，对题中表 1 所给的无人机初始位置的极坐标进行转换，得到附录 4。

7.2.2 位置偏差程度分析和方差分析

为确保我们就发射信号的无人机的位置是无偏差的这一假设不会对最终结果造成剧烈的影响，我们首先对无人机的偏差程度做了一个分析。我们借助 Excel 计算出无人机应抵达的正确坐标，并且加入附录 3 的数据，计算出无人机的偏差距离，再将其与圆形编队的半径（100m）作商，得到附录 5 的数据。其偏差程度是极小的，故我们的假设并不会对最终结果造成结果错误的影响。另外，我们计算出该组数据的方差为 21.49684。若假设方差与相邻两架无人机的距离的平方 r^2 成正比，我们可以得到一个方差关于 r 的估计式：$s^2 = r^2 * 0.00215$，这可以为后面的仿真模拟提供统计学的先验参数。

7.2.3 模型的建立

令 FY00 的坐标为 D，FY01 的坐标为 F，FY0j 的坐标为 B，FY0i 的坐标为 A。假设 FY0i、FY0j 是无偏差的，不难有 AD=DF=DB=R。同时，我们令 $\angle ACD$、$\angle BCD$、$\angle FCD$ 的角度分别为 α_1、α_2、α_3。

如图 7.2 所示，依次做 ΔADC、ΔFDC、ΔBDC 的外接圆 O_1、O_2、O_3。

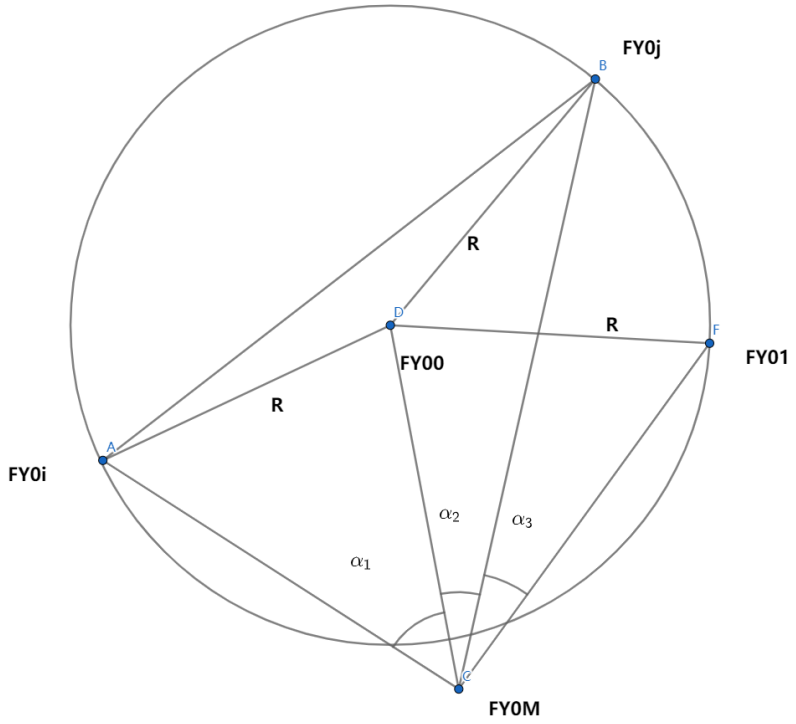


图 7.2：模型的建立

由此，问题转化为通过α_1、α_2、α_3求解O_1、O_2、O_3的方程，进而求解公共的交点。该公共交点实际上不一定存在，但我们可以借助最小二乘法拟合出最优交点，进而确定调整方法。多次重复此过程，即反复迭代，在次数足够多时，无人机可收敛到正确位置。最后，我们通过仿真模拟验证了我们的模型。

7.3 模型的求解

7.3.2 交点的求解

令O_1、O_2、O_3的半径为$R^{(1)}$、$R^{(2)}$、$R^{(3)}$，其圆心的坐标分别为$(x_r^{(1)},y_r^{(1)})$、$(x_r^{(2)},y_r^{(2)})$、$(x_r^{(3)},y_r^{(3)})$。

设$r^{(1)},r^{(2)},r^{(3)}$表示点 x,y 到O_1、O_2、O_3的距离，则有函数关系式如下：

$$\begin{bmatrix} r^{(1)} \\ r^{(2)} \\ r^{(3)} \end{bmatrix} = \begin{bmatrix} \left[(x-x_r^{(1)})^2+(y-y_r^{(1)})^2\right]^{1/2} \\ \left[(x-x_r^{(2)})^2+(y-y_r^{(2)})^2\right]^{1/2} \\ \left[(x-x_r^{(3)})^2+(y-y_r^{(3)})^2\right]^{1/2} \end{bmatrix} = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \\ f_3(x,y) \end{bmatrix} = f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) \quad \cdots\cdots ⑭$$

欲定位飞机 FY0M，由前述可知，通过$\alpha_j^{(1)}$、$\alpha_j^{(2)}$、$\alpha_j^{(3)}$可计算出对应的$R_j^{(k)}$、$x_j^{(k)}$、$y_j^{(k)},(k=1,2,3)$。我们将 r 视为以 x、y 为自变量的二元函数，$x_j^{(k)}$、$y_j^{(k)},(k=1,2,3)$视为参数，原问题即可等价于求解

$$\begin{bmatrix} f_1(x,y) \\ f_2(x,y) \\ f_3(x,y) \end{bmatrix} = \begin{bmatrix} R^{(1)} \\ R^{(2)} \\ R^{(3)} \end{bmatrix} \quad \cdots\cdots ⑮$$

令无人机 i 的理想位置(最终的目标点)为(x_i,y_i)。举例而言，FY01 的理想位置用极坐标表示即为（100，0）。

由于无人机的实际位置距离理想位置很近，我们可以在其理想位置处一阶泰勒展开，将非线性方程的求解转化为线性方程的求解。省略高阶项之后，我们可以得到下式：

$$\begin{bmatrix} r^{(1)}-\hat{r}^{(1)} \\ r^{(2)}-\hat{r}^{(2)} \\ r^{(3)}-\hat{r}^{(3)} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} \end{bmatrix} \begin{bmatrix} x-x \\ y-y \end{bmatrix} \quad \cdots\cdots ⑯$$

其中，$\begin{bmatrix} \hat{r}^{(1)} \\ \hat{r}^{(2)} \\ \hat{r}^{(3)} \end{bmatrix} = f(\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix})$，即无人机在理想位置处到$O_1$、$O_2$、$O_3$的距离可以通过理想位置的坐标带入求解。

为了简化下面的叙述过程，我们记矩阵为 H，即

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} \end{bmatrix} = H = \begin{bmatrix} \frac{x-x_r^{(1)}}{\hat{r}^{(1)}} & \frac{y-y_r^{(1)}}{\hat{r}^{(1)}} \\ \frac{x-x_r^{(2)}}{\hat{r}^{(2)}} & \frac{y-y_r^{(2)}}{\hat{r}^{(2)}} \\ \frac{x-x_r^{(3)}}{\hat{r}^{(3)}} & \frac{y-y_r^{(3)}}{\hat{r}^{(3)}} \end{bmatrix} \quad \cdots\cdots ⑰$$

经过上述处理，H 得到了展开，其值是可求的。

由⑯，对于飞机 FY0M，我们有下式

$$\begin{bmatrix} r_M^{(1)}-\hat{r}_M^{(1)} \\ r_M^{(2)}-\hat{r}_M^{(2)} \\ r_M^{(3)}-\hat{r}_M^{(3)} \end{bmatrix} = H \begin{bmatrix} x_M-x_M \\ y_M-y_M \end{bmatrix} \quad \cdots\cdots ⑱$$

简单来说，无人机在理想位置附近到O_1、O_2、O_3的距离与其横纵坐标呈线性关系，我们可以通过无人机在理想位置到O_1、O_2、O_3的距离$\begin{bmatrix} \hat{r}^{(1)} \\ \hat{r}^{(2)} \\ \hat{r}^{(3)} \end{bmatrix}$和对应的横纵坐标$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$确定线性关系，再带入$\begin{bmatrix} r_M^{(1)} \\ r_M^{(2)} \\ r_M^{(3)} \end{bmatrix} = \begin{bmatrix} R^{(1)} \\ R^{(2)} \\ R^{(3)} \end{bmatrix}$，即能解出其位置。

由于矩阵 H 的行数大于列数，我们可以利用最小二乘法，得到下式

$$\begin{bmatrix} x_j - x_j \\ y_j - y_j \end{bmatrix} = (H^T H)^{-1} H^T \begin{bmatrix} r_M^{(1)} - \hat{r}_M^{(1)} \\ r_M^{(2)} - \hat{r}_M^{(2)} \\ r_M^{(3)} - \hat{r}_M^{(3)} \end{bmatrix}$$

再通过一步变换，我们可以求解出 FY0M 的当前坐标 $\begin{bmatrix} x_M \\ y_M \end{bmatrix}$，即：

$$\begin{bmatrix} x_M \\ y_M \end{bmatrix} = (H^T H)^{-1} H^T \begin{bmatrix} r_M^{(1)} - \hat{r}_M^{(1)} \\ r_M^{(2)} - \hat{r}_M^{(2)} \\ r_M^{(3)} - \hat{r}_M^{(3)} \end{bmatrix} + \begin{bmatrix} x_M \\ y_M \end{bmatrix} \quad \cdots\cdots ⑲$$

这时，代入⑰，无人机即可得到自己的相对于正确位置的距离，并借助自身的行进系统校正自己的位置。

7.4 模型的验证

最后，我们借助 Python 语言，设计了模拟仿真代码，即附录 6。如图 7.3 所示，经过数十次迭代后，我们已经可以认定无人机回归到了正确的位置。由此，我们也可以看出，误差与迭代次数是呈指数级下降的。在实际应用的过程中，我们可以根据我们实际所需要的精度，对迭代次数进行适当的调整。

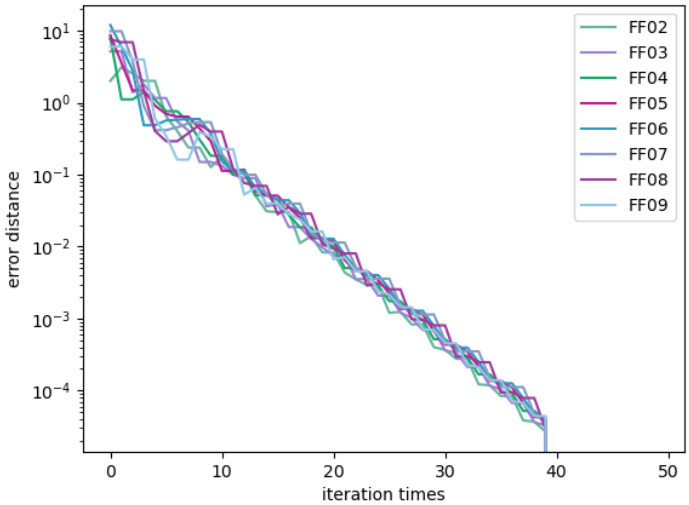


图 7.3 迭代次数与误差之间的关系

八：问题二的模型建立与求解

8.1 问题分析

问题二要求我们同样在纯方位无源定位的情况下，设计锥形编队的无人机位置调整方案。相比于第一题，问题二缺乏一些关键的信息。因此在做第二问

之前，除去与第一问共用的部分基本假设，我们还需要自行做出额外的假设。此外，问题二采取的是锥形编队，因此对我们所设计的方法而言，其需要对无人机关于无人机位置做出一定的分类，才能使所有的无人机抵达正确的位点。

在计算过程中，本题我们仍然假设所有发射信号的无人机都是无偏差的，且需要调整的无人机是略有偏差的。我们所使用的数据方差的规律大致与第一题第（3）问的水平相同。此外，我们以 FY13 为原点，FY13 指向 FY01 作为 X 轴正方向，FY13 指向 FY11 作为 Y 轴正方向，构建了直角坐标系。

我们将调整方案的一轮分为两步。第一步，我们选中 FY05、FY08、FY09 作为发出信号的无人机。对于所有无人机而言，存在如 FY02 这类与中心三架无人机共线的无人机以及 FY01、FY15 这类作为基准无需校正的无人机。因此，我们将关于无人机的位置，对无人机进行分类讨论。与第一题第（3）问类似地，做三点共圆，并据上述数据，使用最小二乘法与构建 TayIor 级数展开定位模型，得到受信号无人机的一个“伪”坐标。且由于我们构建了直角坐标系。无人机可以借助“伪”坐标与正确坐标，前往距离正确目标更近的中间目标。第二步，我们选中 FY01、FY11、FY15 三架无人机发射信号，同样寻找到一个正确的目标。重复上述过程，多次进行迭代调整，无人机最终将逐渐收敛至自己的正确位点。

8.2 模型假设

1. 所需要进行位置调整的无人机是略有偏差的，而非完全背离原应位置；
2. 各个无人机在同一高度飞行；
3. 无人机严格按照题设排序，包括相对位置不变。
4. FY01 与 FY15 的位置是无偏差的。
5. 在标准情况下，无人机之间的距离为 50m。
6. 它们的偏差 Δx 满足 $\Delta x \sim N(0,\delta^2)$ 。

8.3 模型求解与验证

8.3.1 无人机分类

根据前文的分析，我们对无人机做出分类。我们定义集合 A 为特殊调整无人机，且 A 中无人机是与 FY05、FY08、FY09 可能存在共线情况的无人机。那么我们易得：

$$A=\{FY02, FY03, FY07, FY10, FY12, FY14\}$$

接着，由于 FY01 与 FY15 是我们的参考无人机，因此，它们不需要调整。我们定义集合 C 为无需调整无人机，即有:

$$C=\{FY01,FY15\}$$

最后，我们定义除 A 与 C 外的无人机为正常调整无人机，显然有集合 B 为：

$$B=\{FY04, FY05, FY06, FY08, FY09, FY11, FY13\}$$

8.3.2 模型的建立

由于不同位置的飞机需要采取不同的方法处理，因此，以下我们将分类展开讨论。

8.3.2.1 A 类无人机调整方案

由于有三架无人机发出信号，那么接受信号的无人机将接收到三个角度信号。

如图 8.1 所示，若其有一个角度足够小甚至接近于 0，那么我们可以断定这个无人机出现了三点共线或者近似三点共线的情况。此时我们将只做出两个圆，以这两个圆的交点作为它的伪坐标；若其不三点共线，此时，我们可以将其当成普通的情况来处理。普通情况的处理与我们即将探讨的 8.3.2.2 完全一致。

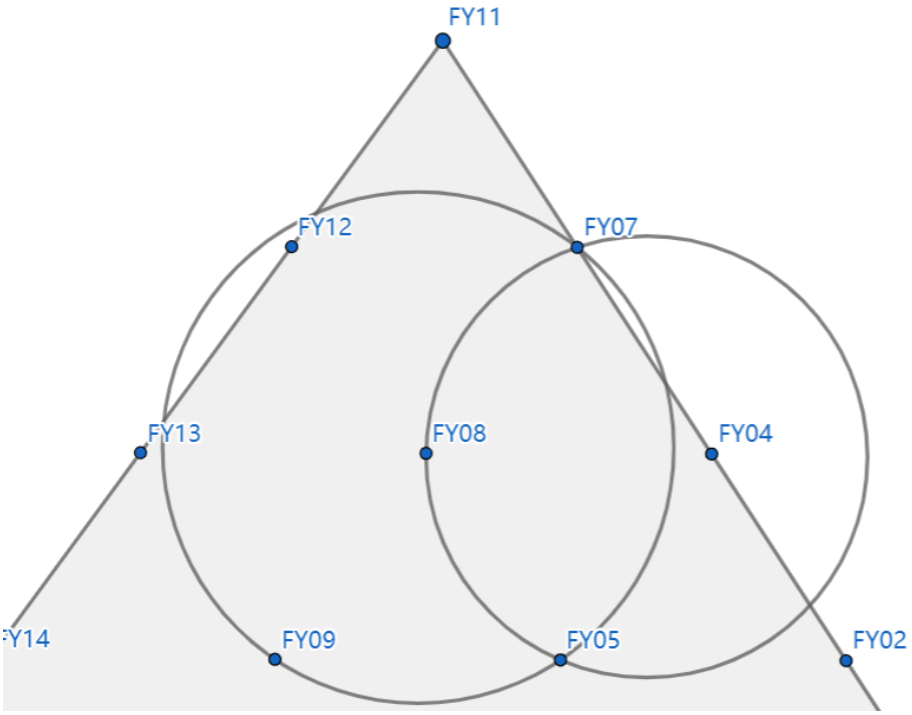


图 8.1：三点共线情况下二圆共点

8.3.2.2 B 类无人机调整方案

此类无人机的调整，与问题一第（3）问完全一致，因此，我们直接借用 7.3.2 中的公式⑰与⑲，对其伪坐标进行求解。

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} \end{bmatrix} = H = \begin{bmatrix} \frac{x-x_r^{(1)}}{\hat{r}^{(1)}} & \frac{y-y_r^{(1)}}{\hat{r}^{(1)}} \\ \frac{x-x_r^{(2)}}{\hat{r}^{(2)}} & \frac{y-y_r^{(2)}}{\hat{r}^{(2)}} \\ \frac{x-x_r^{(3)}}{\hat{r}^{(3)}} & \frac{y-y_r^{(3)}}{\hat{r}^{(3)}} \end{bmatrix} \quad \cdots\cdots ⑰$$

$$\begin{bmatrix} x_M \\ y_M \end{bmatrix} = (H^T H)^{-1} H^T \begin{bmatrix} r_M^{(1)} - \hat{r}_M^{(1)} \\ r_M^{(2)} - \hat{r}_M^{(2)} \\ r_M^{(3)} - \hat{r}_M^{(3)} \end{bmatrix} + \begin{bmatrix} x_M \\ y_M \end{bmatrix} \quad \cdots\cdots ⑲$$

8.3.2.3 C 类无人机调整方案

C 类无人机作为我们调整的参考基准，其既没有接受调整的必要性，也没有接受调整的可行性，因此，我们在仿真模拟的过程中，无需对其进行调整。

8.3.3 模型的仿真模拟

由 7.2.2，我们可以估计出半径为 50m 的情况下，横纵坐标的误差服从均值为零，标准差为 2.31 的正态分布，且它们之间相互独立。同样借助 Python 语言，我们设计了模拟仿真代码，即附录 7。如图 8.2 与图 8.3 所示，经过十至二十次迭代后，我们发现平均误差已经小于 0.1m，可以认定无人机回归到了正确的位置。另外，我们也可以看出，误差与迭代次数是呈指数级下降的。在实际应用的过程中，我们可以根据我们实际所需要的精度，对迭代次数进行适当的调整。

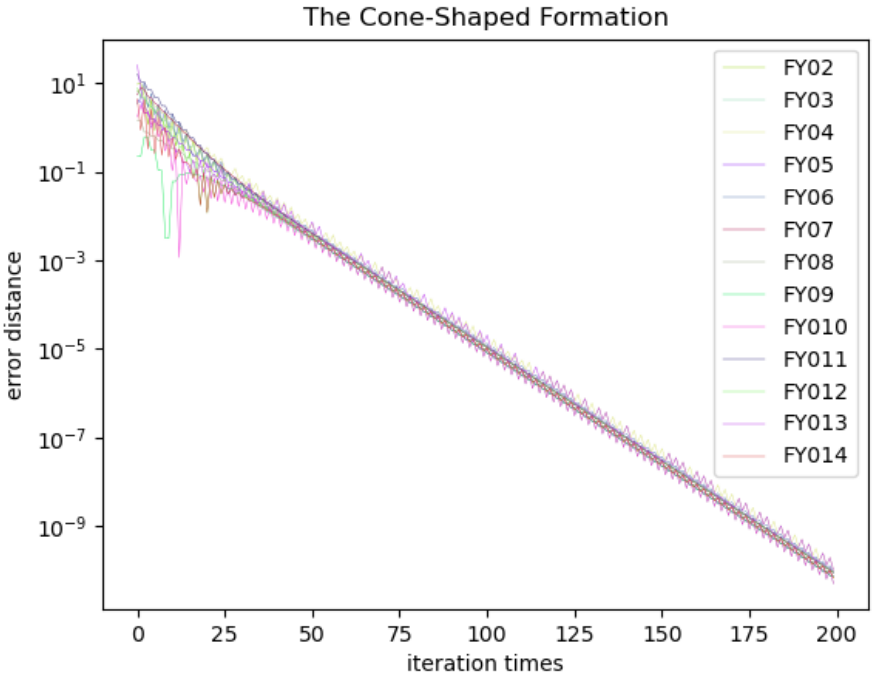


图 8.2 各无人机偏差距离与迭代次数之间的关系

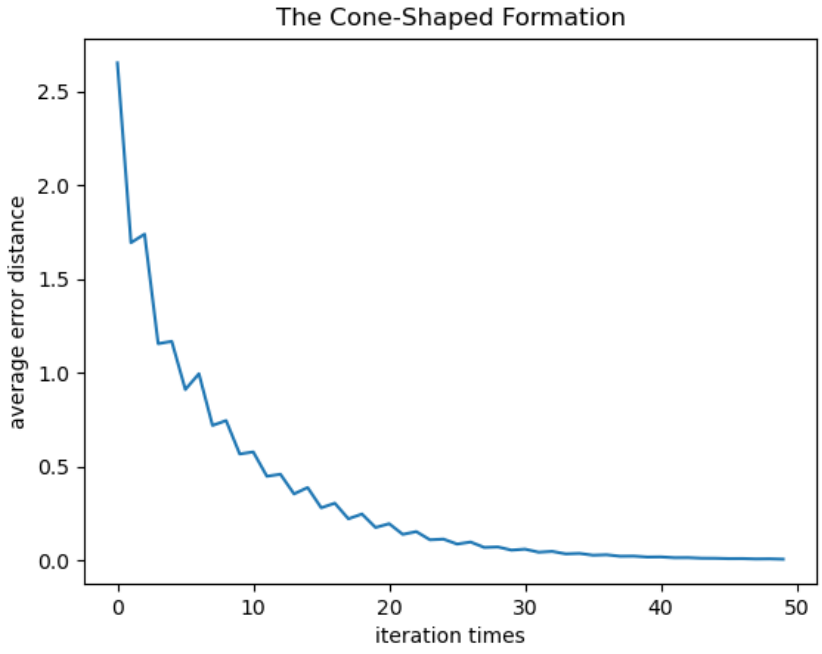
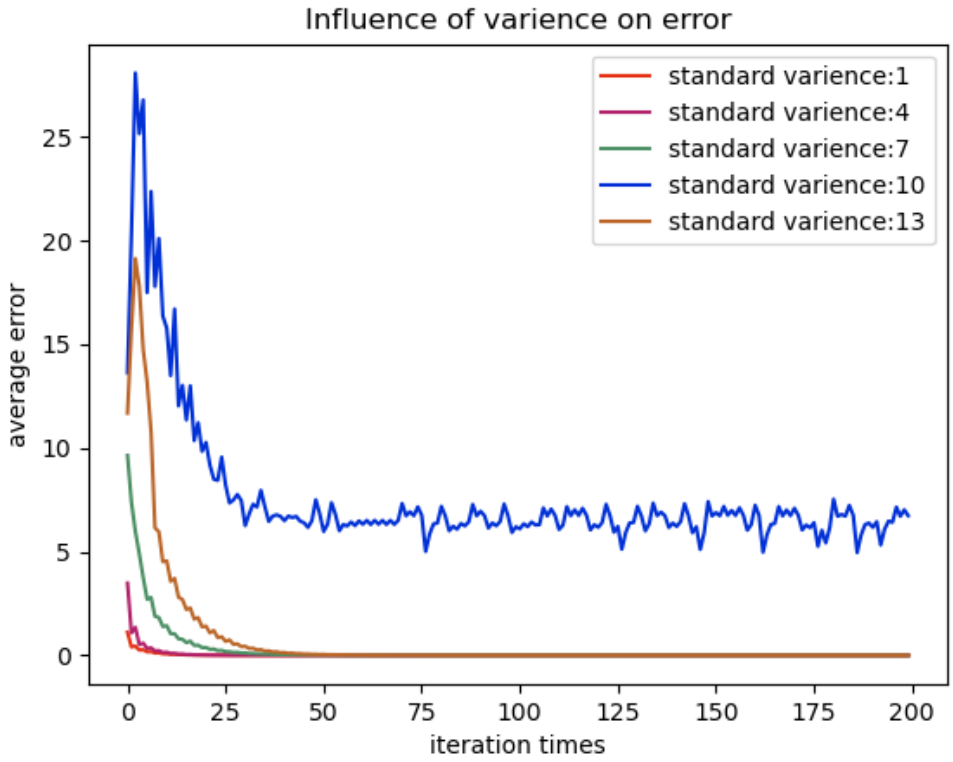


图 8.3 各无人机平均偏差距离与迭代次数之间的关系

8.3.4 方差对平均偏差距离与迭代次数之间的影响

对于偏差距离的标准差不同的情况，我们也进行了多次的仿真模拟。结果表明，当标准差不大于 10 时，我们的算法能给出有效的调整方案，使得所有的无人机都能够调整至正确位置；在标准差大于 10 的情况下，可能会有部分无人机无法调整回正确位置，但仍大部分无人机能够归位。此外，我们还能发现，标准差越小，误差收敛于零的速率越快。下面是一次仿真模拟的结果：



九：模型的评价

9.1 问题一第（1）问模型评价

9.1.1 优点

1. 从几何学角度，严谨、高效地求解了无人机的坐标。
2. 模型的设计及其简洁使用，效率高，为后面的代码计算打下了良好的基础。

9.1.2 缺点

1. 模型基于无人机在相同高度飞行展开，缺乏立体的视角，降低了模型的实用性。
2. 模型要求发射信号的无人机位置无偏差，这限制了其在真实场景中的应用。

9.2 问题一第（2）问模型评价

9.2.1 优点

1. 证明的方法高效简洁，实用性强。合理使用反证法，节省了大量篇幅。
2. 模型考虑的情况全面，避免了特殊情况的出现。
3. 具有代码作为支撑材料，架构严谨。
4. 具有伪代码作为讲解材料，易于理解。

9.2.2 缺点

1. 模型同样基于无人机在相同高度飞行展开，缺乏立体的视角，降低了模型的实用性。

2. 模型没有严谨地证明为何会仅出现有且仅有三圆共点，而是使用了代码计算结果得出的后验经验来说明之。

9.3 问题一第（3）问与问题二模型评价

9.3.1 优点

1.借助代码仿真模拟，具有极高的可靠性

2. 数形结合，既有严谨的几何分析也有高效的数理计算，使模型整体高效实用，收敛性有所保证。

3. 模型精度灵活可控，可以灵活调整迭代次数以调控精度，适用于不同的任务情景，可推广性强。

4. 模型最后给出了偏移程度对调整效果的定量分析，有助于现实场景中提前估计模型效果，进而确定调整方案。

9.3.2 缺点

1. 仅在偏移程度不大的情况适用，若偏移程度足够大，则其可能无法回归至正确位点。

十：参考文献

[1] 秦顾正.对无人机的无源定位关键技术研究[D].东南大学,2019

[2]唐发燕，王运锋.基于 TDOA 的改进定位算法及精度分析[J].现代计算机,2017,(29): 3-8，16

[3] 李瑞雪，夏斌，袁文浩，李彩虹.基于多元变量 Taylor 级数展开模型的定位算法[J].计算机应用研究,2016,第 33 卷(6): 1853-1856，1881

[4]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,(2): 19-21

附录

附录 1：程序预备代码

```
import numpy as np
import math
import matplotlib.pyplot as plt

# 角度制换算到弧度制
def angle_to_radian(x):
    return x/360*2*np.pi

# 极坐标转化为直角坐标
def polar_to_descartes(r, ang):
    rad = angle_to_radian(ang)
    x = r*math.cos(rad)
    y = r*math.sin(rad)
    return x, y

# 计算出两个向量，分别对应于由第一个输入点分别指向后两个输入点
def get_vectors(x, y, z):
    v1 = np.array([y[0]-x[0], y[1]-x[1]])
    v2 = np.array([z[0]-x[0], z[1]-x[1]])
    return v1, v2
# 计算两向量间的弧度制夹角
def radian_between_vectors(x, y):
    Lx = np.sqrt(x.dot(x))
    Ly = np.sqrt(y.dot(y))
    cos_value = x.dot(y)/(Lx*Ly)
    rad=np.arccos(cos_value) #radian
    return rad

#获得由第一个输入点分别指向后两个输入点的两射线间的弧度制夹角
def get_alpha(x, y, z):
    v1, v2 = get_vectors(x, y, z)
    return radian_between_vectors(v1, v2)

# 基于论文 6.2.2.1 部分推导出的公式，基于两点及张角 α，计算出圆心偏下的轨迹圆圆心坐标与半径的平方
def get_low_circle_orbit(p, q, alpha):
    x_o = (p[0]+q[0] + (p[1]-q[1])*math.cos(alpha)/math.sin(alpha))/2
```

```
    y_o = (p[1]+q[1] - (p[0]-q[0])*math.cos(alpha)/math.sin(alpha))/2
    r_square = ((p[0]-q[0])**2 + (p[1]-
q[1])**2)/4/math.sin(alpha)/math.sin(alpha)
    o = (x_o, y_o)
    return o, r_square

# 同基于论文 6.2.2.1 部分推导出的公式，基于两点及张角 α，计算出圆心偏上的轨迹圆圆心坐标与半径的平方
def get_high_circle_orbit(p, q, alpha):
    x_o = (p[0]+q[0] - (p[1]-q[1])*math.cos(alpha)/math.sin(alpha))/2
    y_o = (p[1]+q[1] + (p[0]-q[0])*math.cos(alpha)/math.sin(alpha))/2
    r_square = ((p[0]-q[0])**2 + (p[1]-
q[1])**2)/4/math.sin(alpha)/math.sin(alpha)
    o = (x_o, y_o)
    return o, r_square

# 由已知点 x 位于实际轨迹圆上，在圆心偏下的轨迹圆和圆心偏上的轨迹圆中选出实际的轨迹圆，返回其半径和圆心的横纵坐标
def get_single_circle(x, p, q, alpha):
    o1, r_square = get_low_circle_orbit(p, q, alpha)
    o2, r_square = get_high_circle_orbit(p, q, alpha)

    d1_square = ((x[0]-o1[0])**2)+((x[1]-o1[1])**2)
    d2_square = ((x[0]-o2[0])**2)+((x[1]-o2[1])**2)

    k1 = math.fabs(d1_square - r_square)
    k2 = math.fabs(d2_square - r_square)
    r = math.sqrt(r_square)
    if k1<k2:
        return r, o1[0], o1[1]
    else:
        return r, o2[0], o2[1]

# 输入两圆圆心 p1,p2 及相应的半径 r1,r2,返回非原点的一个交点坐标，为第一题第二问中求三圆交点做准备
def insection_of_circles(p1,r1,p2,r2):
    d = math.sqrt((abs(p2[0]-p1[0]))**2 + (abs(p2[1]-p1[1]))**2)
    if d > (r1+r2) or d < (abs(r1-r2)):
        # print ("两圆不相交")
        return
    elif d == 0 and r1 == r2 :
        # print ("两圆圆心相同")
        return
    else:
```

```
        A = (r1**2 - r2**2 + d**2) / (2 * d)
        if r1**2 - A**2<0:
            return
        h = math.sqrt(r1**2 - A**2)
        x2 = p1[0] + A * (p2[0]-p1[0])/d
        y2 = p1[1] + A * (p2[1]-p1[1])/d
        x3 = round(x2 - h * (p2[1] - p1[1]) / d,8)
        y3 = round(y2 + h * (p2[0] - p1[0]) / d,8)
        if x3 != y3:
            return [x3, y3]
        x4 = round(x2 + h * (p2[1] - p1[1]) / d,8)
        y4 = round(y2 - h * (p2[0] - p1[0]) / d,8)
        return [x4, y4]

# 创建 ndarray 类型，保存直角坐标系下第一题中圆形编队各位点的理想坐标，索引与无人机编号相对应

fy_right = [np.array([0, 0])]
for i in range(9):
    fy_right.append(np.array(polar_to_descartes(100, 40*i)))
fy_right = np.array(fy_right)
```

附录 2：第一题第（2）问的代码 1

```
# 随机生成待调整无人机编号,基于标准正态分布随机分配其位置;整个图的等比例放大缩小不会影响角度，我们假设编队半径为 100m
m = np.random.randint(2,10)
delta_x = np.random.standard_normal(1)[0]
delta_y = np.random.standard_normal(1)[0]
m_position = np.array([fy_right[m][0]+delta_x, fy_right[m][1]+delta_y])

# 计算 FY00，FY01 与生成的待调整无人机之间的张角 α1 大小与轨迹圆
alpha1 = get_alpha(m_position, fy_right[0], fy_right[1])
r1, o1_x, o1_y = get_single_circle(m_position, fy_right[0],
fy_right[1], alpha1)
o1 = np.array([o1_x, o1_y])

# 随机为未知编号的发射信号无人机指定编号为 i_real,便于接下来的讨论
i_real = np.random.randint(2,10)
while(i_real == m):
    i_real = np.random.randint(2,10)
alpha2 = get_alpha(m_position, fy_right[0], fy_right[i_real])
alpha3 = get_alpha(m_position, fy_right[1], fy_right[i_real])
```

```
# 接收信号的无人机基于所接收到的三个 α 角，遍历假设未知编号的发射信号无人机的所有可行编号，
# 分别计算出两个新圆 2，3；并求圆 1 与圆 2、3 分别的交点；若三圆共交于一点，将该点添加到结果列表 A 中
A = []
for i in range(2, 10):
    if i == m:
        continue
    r2, o2_x, o2_y = get_single_circle(fy_right[m], fy_right[0],
fy_right[i], alpha2)
    o2 = np.array([o2_x, o2_y])
    r3, o3_x, o3_y = get_single_circle(fy_right[m], fy_right[1],
fy_right[i], alpha3)
    o3 = np.array([o3_x, o3_y])

    insec_12 = insection_of_circles(o1, r1, o2, r2)
    insec_13 = insection_of_circles(o1, r1, o3, r3)

    if (insec_12 is not None) and (insec_13 is not None):
        if math.fabs(insec_12[0]-insec_13[0])<1e-1 and
math.fabs(insec_12[1]-insec_13[1])<1e-1 and insec_12[0]!=100 and
math.fabs(insec_12[0]-0)>1:
            A.append(insec_12)

print('待调整无人机为 FY0{},位置为{};\n 未知编号的无人机实际为 FY0{},\n 此时，得到的可行三圆交点为{}'.format(m, m_position, i_real, A))
if A==[]:
    print("无有效交点,无法定位")
if len(A)>=2:
    print("输出多个交点,无法有效定位")
```

附录 3：第一题第（2）问的代码 2

```
# 随机生成待调整无人机编号及位置
m = np.random.randint(2,10)
delta_x = np.random.standard_normal(1)[0]
delta_y = np.random.standard_normal(1)[0]
m_position = np.array([fy_right[m][0]+delta_x, fy_right[m][1]+delta_y])

# 计算 FY00，FY01 与生成的待调整无人机之间的张角 α1 大小与轨迹圆
alpha1 = get_alpha(m_position, fy_right[0], fy_right[1])
r1, o1_x, o1_y = get_single_circle(m_position, fy_right[0],
fy_right[1], alpha1)
o1 = np.array([o1_x, o1_y])
```

```
# 随机为未知编号的第一架发射信号无人机指定编号为 i_real
i_real = np.random.randint(2,10)
while(i_real == m):
    i_real = np.random.randint(2,10)
alpha2 = get_alpha(m_position, fy_right[0], fy_right[i_real])

# 随机为未知编号的第二架发射信号无人机指定编号为 j_real
j_real = np.random.randint(2,10)
while(j_real == m or j_real == i_real):
    j_real = np.random.randint(2,10)
alpha3 = get_alpha(m_position, fy_right[0], fy_right[j_real])

# 接收信号的无人机基于所接收到的三个 α 角，二重循环遍历所有未知编号的两架发射信号无人机的所有可行编号组合
# 基于此分别计算出两个新圆 2，3；并求圆 1 与圆 2、3 分别的交点；若三圆共交于一点，将该点添加到结果列表 A 中
A = []
for i in range(2, 10):
    if i == m:
        continue
    for j in range(2, 10):
        if j == m or j == i:
            continue
        r2, o2_x, o2_y = get_single_circle(fy_right[m], fy_right[0],
fy_right[i], alpha2)
        o2 = np.array([o2_x, o2_y])
        r3, o3_x, o3_y = get_single_circle(fy_right[m], fy_right[0],
fy_right[j], alpha3)
        o3 = np.array([o3_x, o3_y])
        insec_12 = insection_of_circles(o1, r1, o2, r2)
        insec_13 = insection_of_circles(o1, r1, o3, r3)

        if (insec_12 is not None) and (insec_13 is not None):
            if math.fabs(insec_12[0]-insec_13[0])<1e-6 and
math.fabs(insec_12[1]-insec_13[1])<1e-6 and insec_12[0]!=100 and
math.fabs(insec_12[0]-0)>1:
                A.append(insec_12)

print('待调整无人机为 FY0{},位置为{};\n 未知编号的两架无人机实际为 FY0{}和 FY0{},\n 此时,得到的可行三圆交点为{},实现了有效定位'.format(m, m_position,
i_real, j_real, A))
```

附录 4：第一题第（3）问无人机初始位置的直角坐标表示

无人机编号	直角坐标（x,y)

0	(0, 0)
1	(100, 0)
2	(74.96, 63.12)
3	(19.04, 110.37)
4	(-52.10, 91.16)
5	(-92.01, 33.74)
6	(-105.27, -38.23)
7	(-52.39, -91.00)
8	(17.30, -96.46)
9	(86.15, -71.57)

附录 5：第一题第（3）问无人机位置偏差程度分析

无人机编号	偏差距离（单位：米）	偏差程度
0	0.00	0.00%
1	0.00	0.00%
2	2.01	2.01%
3	12.01	12.01%
4	5.02	5.02%
5	2.01	2.01%
6	12.00	12.00%
7	5.00	5.00%
8	2.02	2.02%
9	12.01	12.01%

附录 6：第一题第（3）问的代码

```
# 第三问中实际的各无人机坐标的初始化,保存到 ndarray 中,索引与无人机编号相对应
fy00 = np.array([0, 0])
fy01 = np.array([100, 0])
fy02 = np.array(polar_to_descartes(98, 40.10))
fy03 = np.array(polar_to_descartes(112, 80.21))
fy04 = np.array(polar_to_descartes(105, 119.75))
fy05 = np.array(polar_to_descartes(98, 159.86))
fy06 = np.array(polar_to_descartes(112, 199.96))
fy07 = np.array(polar_to_descartes(105, 240.07))
fy08 = np.array(polar_to_descartes(98, 280.17))
fy09 = np.array(polar_to_descartes(112, 320.28))
fy = np.array([fy00, fy01, fy02, fy03, fy04, fy05, fy06, fy07, fy08,
fy09])

# 第 1 个变量为待调整无人机编号 第 2、3、4 个变量为除 FY00 外发射信号的无人机编
号，分别计算待调整无人机与 FY00，输入编号的无人机的张角，输出三个 α 构成的 array
def get_alphas(x, a, b, c):
    x = fy[x]
```

```
    o = fy[0]
    a = fy[a]
    b = fy[b]
    c = fy[c]
    alpha1 = get_alpha(x, o, a)
    alpha2 = get_alpha(x, o, b)
    alpha3 = get_alpha(x, o, c)
    alpha = np.array([alpha1, alpha2, alpha3])
    return alpha

# 输入各发射信号的无人机编号，以 ndarray 的形式返回它们分别的理想点位构成的矩阵
def get_ideal_positions(a, b, c):
    a = fy_right[a]
    b = fy_right[b]
    c = fy_right[c]
    return np.array([a, b, c])

# 输入各发射信号的无人机编号，以 ndarray 的形式返回它们分别的真实位置构成的矩阵
def get_real_positions(a, b, c):
    a = fy[a]
    b = fy[b]
    c = fy[c]
    return np.array([a, b, c])

#输入待调整无人机编号，除 FY00 外发射信号的各无人机编号，以 ndarray 的形式返回
与角度向量顺序一致的三个圆分别的[半径平方，圆心横坐标，圆心纵坐标]构成的矩阵
def get_circles(x, a, b, c):
    o = fy[0]
    alpha = get_alphas(x, a, b, c)
    ideal_position = get_ideal_positions(a, b, c)
    circle1 = get_single_circle(fy_right[x], o, ideal_position[0],
alpha[0])
    circle2 = get_single_circle(fy_right[x], o, ideal_position[1],
alpha[1])
    circle3 = get_single_circle(fy_right[x], o, ideal_position[2],
alpha[2])
    return np.array([circle1, circle2, circle3])

iter_times=10
errors = np.zeros((iter_times * 5, 10))
error_2=np.zeros((iter_times*4,2))
def circle_adjust(k):
    global errors
    for i in range(2, 6):
```

```
        mov = np.zeros((10, 2))
        for j in range(2, 10):
            if j == i or j == i + 4:
                continue
            # 获得三个 alpha
            alpha = get_alphas(j, 1, i, i + 4)
            # 获得三组圆
            circles = get_circles(j, 1, i, i + 4)
            circles_r = circles[:, 0]
            circles_xy = circles[:, 1:]

            r_ideal = np.sqrt(np.sum((circles_xy - fy_right[j]) ** 2,
axis=1))
            H = -(circles_xy - fy_right[j]) / r_ideal.reshape((-1, 1))

            ans = np.linalg.inv(H.T @ H) @ H.T @ (circles_r - r_ideal)
            mov[j] = ans
            print("FY{},{}发射信号,FY{}计算得到其位移为{}".format(i, i +
4, j, mov[j]))
        global fy
        fy -= mov
        print("第{}圈调整后".format(4 * k + i - 2))
        for l in range(10):
            print("FY{}的位置偏差为: ".format(l), fy_right[l][0] -
fy[l][0], fy_right[l][1] - fy[l][1])
        errors[4 * k + i - 2,:] = np.sqrt(np.sum((fy_right - fy)** 2,
axis=1))
        error_2[4 * k + i - 2,:]=(fy_right[2] - fy[2])

for i in range(iter_times):
    circle_adjust(i)

plt.figure()
for i in range(2, 10):
    plt.yscale('log')
    plt.plot(errors[:,
i],linewidth=0.3,color=(np.random.uniform(),np.random.uniform(),np.rand
om.uniform()),label="FY0{}".format(i))

plt.legend()
plt.xlabel('iteration times')
plt.ylabel('error distance')
plt.savefig("all_errors.png")
```

附录 7：第二题的代码

```
# 第二题中实际的各无人机坐标的初始化,保存到 ndarray 中,(索引＋1)与无人机编号对应
fy13 = np.array([0, 0])
fy12 = np.array([0, 50])
fy11 = np.array([0, 100])
fy14 = np.array([0, -50])
fy15 = np.array([0, -100])
fy08 = np.array([50*math.sin(math.pi/3), 25])
fy07 = np.array([50*math.sin(math.pi/3), 75])
fy09 = np.array([50*math.sin(math.pi/3), -25])
fy10 = np.array([50*math.sin(math.pi/3), -75])
fy05 = np.array([2*50*math.sin(math.pi/3), 0])
fy04 = np.array([2*50*math.sin(math.pi/3), 50])
fy06 = np.array([2*50*math.sin(math.pi/3), -50])
fy02 = np.array([3*50*math.sin(math.pi/3), 25])
fy03 = np.array([3*50*math.sin(math.pi/3), -25])
fy01 = np.array([4*50*math.sin(math.pi/3), 0])

fy_cone_right =
np.array([fy01,fy02,fy03,fy04,fy05,fy06,fy07,fy08,fy09,fy10,fy11,fy12,fy13,fy14,fy15])

# 基于计算得到合理标准差的正态分布，随机初始化第二题中实际的各无人机坐标,保存到 ndarray 中,(索引＋1)与无人机编号对应
fy_cone =
np.array([fy01,fy02,fy03,fy04,fy05,fy06,fy07,fy08,fy09,fy10,fy11,fy12,fy13,fy14,fy15])
fy_cone[0] = fy_cone_right[0]
fy_cone[14] = fy_cone_right[14]

std_variance=4

for i in range(1, 14):
    delta_x = np.random.standard_normal(1)[0]*std_variance
    delta_y = np.random.standard_normal(1)[0]*std_variance
    fy_cone[i] = np.array([fy_cone_right[i][0]+delta_x,
fy_cone_right[i][1]+delta_y])

# 第 1 个变量为待调整无人机编号,第 2、3、4 个变量为发射信号的无人机编号，即为 a,b,c;
# 分别计算待调整无人机与 a,b;a,c;b,c 三组发射信号的无人机的张角，输出三个 α 构成的 array
def get_cone_alphas(x, a, b, c):
```

```
    x = fy_cone[x-1]
    a = fy_cone[a-1]
    b = fy_cone[b-1]
    c = fy_cone[c-1]
    alpha1 = get_alpha(x, a, b)
    alpha2 = get_alpha(x, a, c)
    alpha3 = get_alpha(x, b, c)
    alpha = np.array([alpha1, alpha2, alpha3])
    return alpha

# 输入各发射信号的无人机编号，以 ndarray 的形式返回它们分别的理想点位构成的矩阵
def get_cone_ideal_positions(a, b, c):
    a = fy_cone_right[a-1]
    b = fy_cone_right[b-1]
    c = fy_cone_right[c-1]
    return np.array([a, b, c])

# 输入各发射信号的无人机编号，以 ndarray 的形式返回它们分别的真实位置构成的矩阵
def get_cone_real_positions(a, b, c):
    a = fy_cone[a-1]
    b = fy_cone[b-1]
    c = fy_cone[c-1]
    return np.array([a, b, c])

#输入待调整无人机编号，发射信号的各无人机编号，以 ndarray 的形式返回与角度向量顺序一致的三个圆分别的[半径平方，圆心横坐标，圆心纵坐标]构成的矩阵
def get_cone_circles(x, a, b, c):
    alpha = get_cone_alphas(x, a, b, c)
    ideal_position = get_cone_ideal_positions(a, b, c)
    circle1 = get_single_circle(fy_cone_right[x-1], ideal_position[0], ideal_position[1], alpha[0])
    circle2 = get_single_circle(fy_cone_right[x-1], ideal_position[0], ideal_position[2], alpha[1])
    circle3 = get_single_circle(fy_cone_right[x-1], ideal_position[1], ideal_position[2], alpha[2])
    return np.array([circle1, circle2, circle3])

iter_time2=100
cone_errors=np.zeros((iter_time2*2,15))

fy_senders = np.array([[1,11,15],[5,8,9]])
def cone_adjust(k):
    global fy_cone
    for i in range(2):
```

```
        fy_sender = fy_senders[i]
        mov=np.zeros((15,2))
        for j in range(2, 15):
            if j in fy_sender:
                continue
            # 获得三个 alpha
            alpha = get_cone_alphas(j, fy_sender[0], fy_sender[1],
fy_sender[2])
            # 获得三组圆
            circles = get_cone_circles(j, fy_sender[0], fy_sender[1],
fy_sender[2])
            circles=circles[circles[:,0]<500]
            circles_r = circles[:, 0]
            circles_xy = circles[:, 1:]

            r_ideal = np.sqrt(np.sum((circles_xy - fy_cone_right[j-1])
** 2, axis=1))
            H = -(circles_xy - fy_cone_right[j - 1]) /
r_ideal.reshape((-1, 1))
            ans = np.linalg.inv(H.T @ H) @ H.T @ (circles_r - r_ideal)
            mov[j-1] = ans
            print("FY{}发射信号,FY{}计算得到其位移为{}".format(fy_sender,
j, -mov[j-1]))
        fy_cone-=mov
        print("第{}圈调整后".format(3 * k + i))
        for l in range(15):
            print("FY{}的位置偏差为: ".format(l+1), fy_cone_right[l][0] -
fy_cone[l][0], fy_cone_right[l][1] - fy_cone[l][1])
        cone_errors[2*k+i,:]=np.sqrt(np.sum((fy_cone_right-
fy_cone)**2,axis=1))

for i in range(100):
    cone_adjust(i)

plt.figure()
for i in range(1, 14):
    plt.yscale('log')
    plt.plot(cone_errors[:,
i],linewidth=0.3,color=(np.random.uniform(),np.random.uniform(),np.rand
om.uniform()),label="FY0{}".format(i+1))
plt.legend()
plt.xlabel('iteration times')
plt.ylabel('error distance')
plt.savefig("all_errors_p2.png")
```

方差对误差的影响探究代码

```
iter_time2 = 100
std_variance1 = 10
std_variance2=std_variance1/2
std_variance = std_variance2
variance_cnt=26
std_variances = np.linspace(0, 25, variance_cnt)

cone_errors=np.zeros((iter_time2*2,15))
variance_k_averror=np.zeros((variance_cnt,iter_time2*2))
# 方差对误差的影响
fy_senders = np.array([[1,11,15],[5,8,9]])
def cone_adjust(k,vidx=0):
    global fy_cone,cone_errors
    for i in range(2):
        fy_sender = fy_senders[i]
        mov=np.zeros((15,2))
        for j in range(2, 15):
            if j in fy_sender:
                continue
            # 获得三个 alpha
            alpha = get_cone_alphas(j, fy_sender[0], fy_sender[1],
fy_sender[2])
            # 获得三组圆
            circles = get_cone_circles(j, fy_sender[0], fy_sender[1],
fy_sender[2])
            circles = circles[circles[:, 0] < 500]
            m, n = circles.shape
            if m>1:
                circles_r = circles[:, 0]
                circles_xy = circles[:, 1:]
                r_ideal = np.sqrt(np.sum((circles_xy - fy_cone_right[j-
1]) ** 2, axis=1))
                H = -(circles_xy - fy_cone_right[j - 1]) /
r_ideal.reshape((-1, 1))
                ans = np.linalg.inv(H.T @ H) @ H.T @ (circles_r -
r_ideal)
                mov[j-1] = ans
            #print("FY{}发射信号,FY{}计算得到其位移为{}".format(fy_sender,
j, -mov[j-1]))
        fy_cone-=mov
        #print("第{}圈调整后".format(3 * k + i))
        # for l in range(15):
```

```
        #     print("FY{}的位置偏差为: ".format(l+1), fy_cone_right[l][0]
- fy_cone[l][0], fy_cone_right[l][1] - fy_cone[l][1])
        variance_k_averror[vidx,2*k+i]=np.average( np.sqrt(np.sum((fy_co
ne_right-fy_cone)**2,axis=1)))
        #cone_errors[2*k+i,:]=np.sqrt(np.sum((fy_cone_right-
fy_cone)**2,axis=1))
for i0 in range(variance_cnt):
    for i in range(1, 14):
        delta_x = np.random.standard_normal(1)[0]*std_variances[i0]
        delta_y = np.random.standard_normal(1)[0]*std_variances[i0]
        fy_cone[i] = np.array([fy_cone_right[i][0]+delta_x,
fy_cone_right[i][1]+delta_y])
    for i in range(iter_time2):
        cone_adjust(i,i0)

# #所有无人机的误差图
# plt.figure()
# for i in range(1, 14):
#     plt.yscale('log')
#     plt.plot(cone_errors[:,
i],linewidth=0.3,color=(np.random.uniform(),np.random.uniform(),np.rand
om.uniform()),label="FY0{}".format(i+1))
# plt.title("The Cone-Shaped Formation")
# plt.legend()
# plt.xlabel('iteration times')
# plt.ylabel('error distance')
# plt.savefig("all_errors_p2.png")

# # 平均误差图
# plt.figure()
# plt.plot(np.average(cone_errors, axis=1))
# plt.title("The Cone-Shaped Formation")
# plt.xlabel('iteration times')
# plt.ylabel('average error distance')
# plt.savefig("average_errors_p2.png")
print(variance_k_averror)
# x = np.linspace(0, variance_cnt, variance_cnt)
# y=np.linspace(0,iter_time2*2,iter_time2*2)
# Y, X = np.meshgrid(y, x)
# plt.figure()
# fig = plt.figure()
# ax = plt.axes(projection='3d')
# ax.contour3D(X, Y, variance_k_averror)
# plt.savefig("3d.png")
```

```
c1 = [229,204,255]
c1=np.array(c1)
c2 = [0,0,153]
c2=np.array(c2)
plt.figure()
for i in range(1,14,3):
    plt.plot(variance_k_averror[i],label="standard
varience:{}".format(i),color=(np.random.uniform(),np.random.uniform(),n
p.random.uniform()))
plt.legend()
plt.xlabel("iteration times")
plt.ylabel("average error")
plt.title("Influence of varience on error")
plt.savefig("error_varience.png")
```