基于粒子群算法与几何关系的无人机纯方位无源定位算法

摘要

无人机在飞行过程中，为避免干扰，通常需要保持电磁静默。在这种情况下，无人机编队之间的信息交流交换就显得尤为重要。目前为减少电磁干扰，使用最多的技术是**纯方位无源定位技术**，本文围绕运用该技术下的无人机定位的一系列问题展开。

问题一第一小问，通过分析该无人机编队的**几何关系**，通过余弦定理建立角度与边长的关系，用于求解偏离无人机坐标。接着运用模型，对方程组求解。运用**粒子群算法**求解，猜测初始解为原无人机编队中的设定位置，能够大大提升模型求解的精度和准确率。最后，使用随机数模拟偏离无人机位置，测定模型效能。结果显示，在 **1000** 次模拟求解的过程中，求解偏差几乎可以忽略的次数高达 999 次，模型求解准确率为 **0.999**，认为模型效果良好。

问题一第二小问，运用“各无人机相对位置不变”这一规则来进一步确定无人机位置与编号，在第一小问的模型下，增加一个正确位置下各无人机发射信号, 与接收信号得出的**角度矩阵**，将偏离队列无人机所接受到的角度信息，与既有角度信息进行匹配，确定他们的相对位置，继而确定除 FY00、FY01 外的第三架无人机的，编号及位置。模型经过多次模拟验证，认为准确性良好，因此，得出结论除 FY00、FY01 外，至少还需要一架无人机才能有效定位位置。

问题一第三小问，除 FY01 外各无人机位置都有所偏移，采用**分步调整策略**，选取 FY02、FY05、FY07 作为优先调整无人机，依次接收其余无人机信号，并通过角度矩阵区分产生角度的无人机编号，再分为三组采用粒子群算法得出平均相对移动距离进行移动，并循环迭代优化，使三家无人机来到正确位置，再利用问题一模型，将其余无人机调整至正确位置。

问题二，依然采用分布调整，先选取 FY01、FY02、FY03 组成的一个近等边三角形，通过**三角定位**，比较实际角度与 60 度的差距调整无人机位置，使三架无人机形成等边三角形，再通过人为给予偏移量，比较角度变化，利用解析几何算出等边三角形边长，进一步将三架无人机调整至正确位置，再利用正确位置的三架无人机产生的信号依次调节其他无人机，实现位置更正。

关键字： 纯方位无源定位技术； 几何关系； 粒子群算法； 角度矩阵； 分步调整策略; 三角定位;

目录

一、 问题重述

1.1 问题背景

无人机集群在遂行编队飞行时，为避免外界干扰，应尽可能保持电磁静默，少向外发射电磁波信号。在这种情况下，为了保持无人机队形的稳定性，采用纯方位无源定位的方法来保持无人机队形。所谓纯方位无源定位的方法，即由编队中某几架无人机发射信号，其他几架无人机负责接受信号。从中提取方位信息来进行定位。在此基础上，建立合适的数学模型用于无人机定位极为重要。

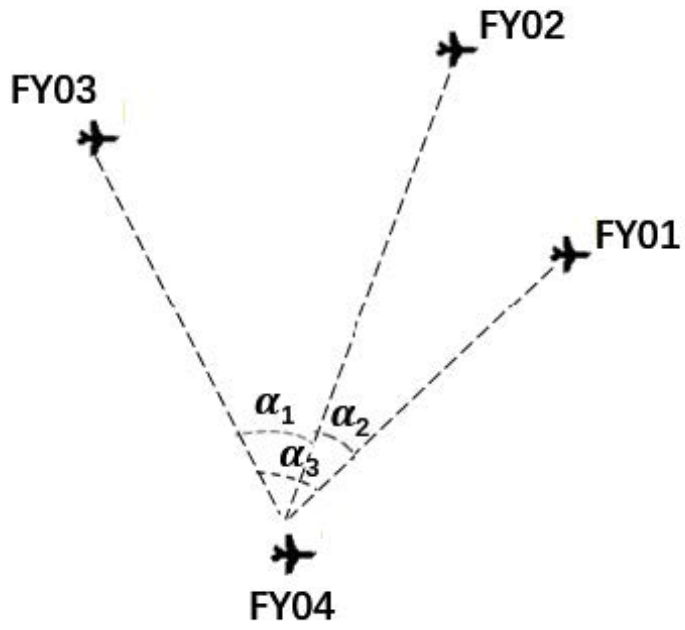


图 **1**　无人机接收到的方向信息示意图

1.2 问题要求

本文围绕无人机纯方位无源定位算法，对无人机位置进行定位与修正的一系列问题的解决来展开。

1.2.1 问题一

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（见图 2）。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。

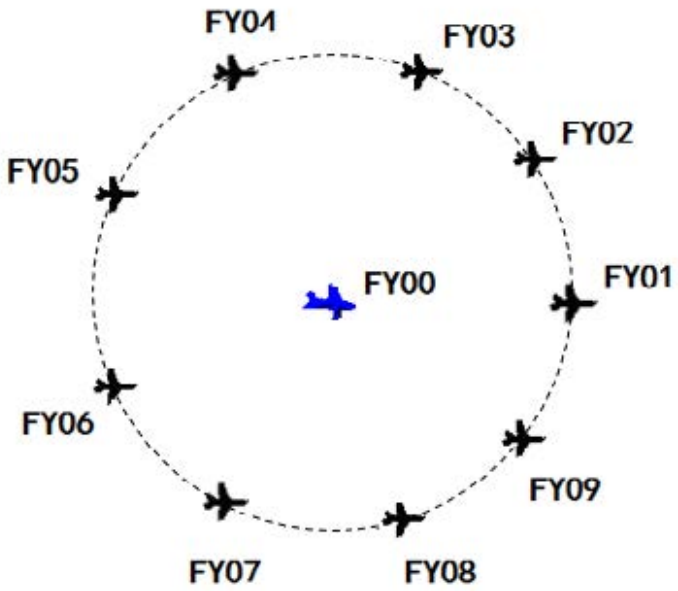


图 **2**　圆形无人机编队示意图

(1) 第一小问：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) 第二小问：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，还需要几架无人机发射信号，才能实现无人机的有效定位？

(3) 第三小问: 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 100 m 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。利用所给数据，给出调整方案。

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（见图 3直线上相邻两架无人机的间距相等如 50 m 。仍考虑纯方位无源定位的情形，设计无人机位置调整方案。

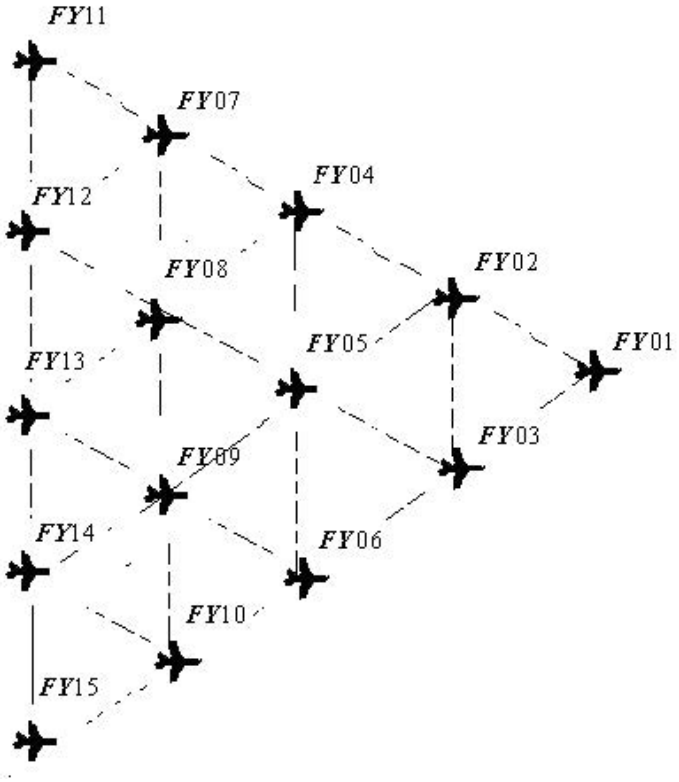


图 **3**　圆形无人机编队示意图

二、 问题分析

2.1 问题一分析

2.1.1 第一小问分析

首先，经过问题分析，需要建立纯方位无源定位的模型，明确输入与输出。

- 明确模型的输入数据：模型输入数据为位于圆心的无人机 FY00 和编队中另 2 架无人机的编号，和三个返回的角度值
- 明确模型需要达到的目的：模型需要根据输入数据，得到无人机的位置信息。
- 确定模型内的几何关系，进行计算。

2.1.2 第二小问分析

首先根据上文分析，建立的方程组至少需要三架无人机给出的定位信息才可以确定未知位置无人机的确定定位。

因此，除了已知 FY00 和 FY01 的位置和编号外，至少还需要知道一架无人机的编号及返回角度值，才可以确定一架无人机的位置。对此，需修改第一小问的模型使模型可以确定无人机的编号。具体流程如图 4

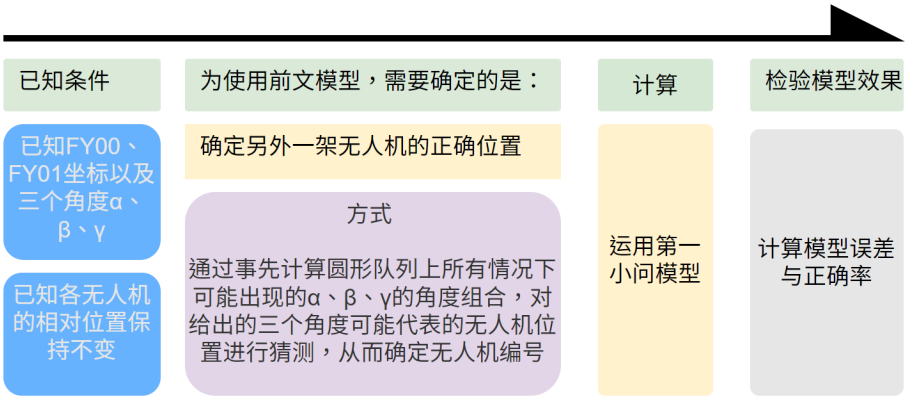


图 4　第二小问流程图

2.1.3 第三小问分析

通过题目所给数据可以看出，第三小问除了 FY00 与 FY01 外其余无人机的位置都发生了略微偏移，对此我们分成两步来解决这一问题。

首先，挑选出三架无人机作为第一步要调整位置的无人机，依次挑选其中一架无人机为接收信号无人机，另外两架与 FY00、FY01 组成发射信号无人机组，并结合角度矩阵进行编号判断与重组，获得三组发射组，每组三个角度，再结合第一问模型，用正确位置代替未知的实际位置，获得相对移动距离并进行多次迭代优化，使挑选出的三架无人机回到正确位置。

接着，结合问题一模型，依次将其余无人机调整至正确位置，无人机的位置调整便完成了。

2.2 问题二分析

这里考虑其他编队队形，但实际与前面相同，仍然是定位调整的问题。本题的无人机位置较前一问不同，若效仿前一问的做法，找一个无人机作为圆心，在圆上找四个无人机来进行迭代调整，由于选圆上的四个无人机必定会有两架无人机的目标位置恰好在圆直径的两端，会导致接受信号时，匹配来源无人机的编号困难，容易与中心无人机匹配错误，使适应度函数计算错误，始终无法收敛。

所以这里考虑利用等边三角形的几何特性，先是挑选三架在等边三角形端点上的无人机互相发送信号，每次会有一架无人机收到另外两个端点上无人机的信号，通过计算角度，调整位置。三架无人机轮流接收信号，迭代调整优化，达到三架无人机恰好形成一个等边三角形。

此时，可以通过其中某架无人机稍微偏移后接收的角度信号，通过解析几何计算得到目前这个等边三角形的边长，从而得到这三架无人机的目前坐标，进而调整到正确位

置。

在已知三架无人机的正确坐标的基础上，在没有三架无人机会在同一直线上的情况下，从未定位的无人机里挑选一台，套用问题一的定位模型，得到目前位置从而调整至正确位置。

三、 模型假设

(1) 无人机发射与接收信号的时间误差可以忽略不记。
(2) 未调整位置时，无人机沿相同速度前进，无人机之间的相对位置不变。
(3) 无人机偏离队伍的误差在一定范围内，不会出现两无人机交换相对位置的情况

四、 符号说明

符号	说明
p_i	单次发射获得的相对位置
p	平均相对位置
Δp	相对移动距离
R_i	编号 FY0i 无人机的正确位置
D_{ij}	编号 FY0i 无人机与编号 FY0j 无人机的欧氏距离
y_i	编号 FY0i 无人机的纵坐标
t	火箭残骸音爆实际发生时刻
t_i	音爆抵达时间
c_1	个体认知加速常数
c_2	社会经验加速常数
$pbest_{i,d}$	粒子 i 在维度 d 上的个体最优位置
$x_{i,d}(t)$	粒子 i 在时间 t 在维度 d 上的当前位置
$gbest_d$	全局最优在维度 d 上的位置

五、 模型的建立与求解

5.1 问题一

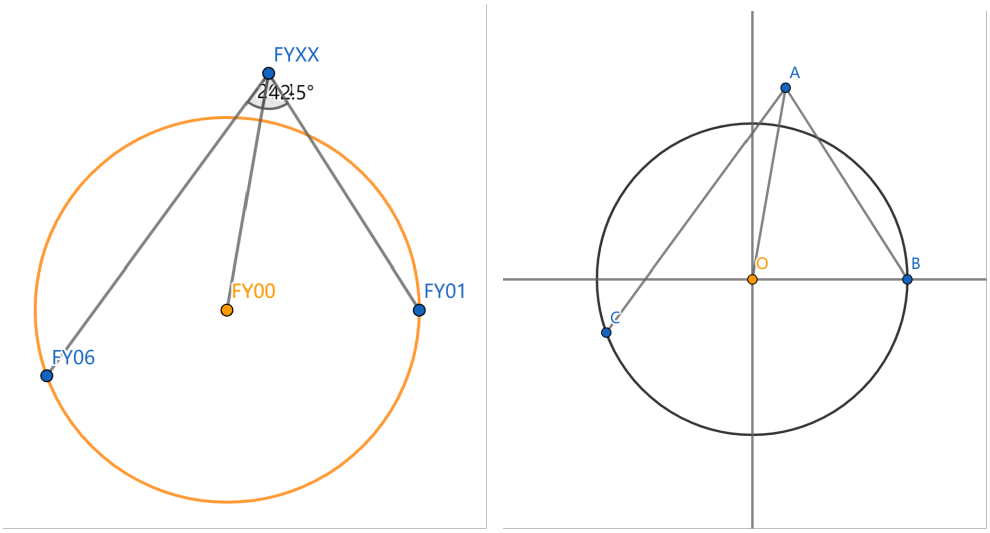
5.1.1 第一小问模型的建立与求解

针对第一小问，我们发现，若已知发射信号无人机编号 FY01 与 FY06，且发射信号无人机位置无偏差，要求解编号为 FYXX 无人机的位置。

- **模型的建立**

据此构建数学模型，将 FY00 无人机视为点 O，将 FY01 无人机视为点 B，将 FY06 无人机视为点 C，将偏移的无人机视为点 A，并以 O 为坐标原点构建坐标系，令 $\angle CAO$ 为 α,$\angle BAO$ 为 β,$\angle CAB$ 为 γ，据余弦定理可得公式（1）, 代入点坐标和角度即可求得 A 点坐标。

$$\begin{cases} \cos\alpha = \frac{(CA^2+AO^2-CO^2)}{2CA\cdot AO} \\ \cos\beta = \frac{(BA^2+AO^2-BO^2)}{2BA\cdot AO} \\ \cos\gamma = \frac{(CA^2+AB^2-CB^2)}{2CA\cdot AB} \end{cases} \tag{1}$$



(a) 模型图示　　(b) 建立坐标系后

图 **5**　模型建立

- **模型的求解**

在模型建立中，我们得到了一个最高次为 2 次，包含两个未知数的方程组，普通求解器在求解方面存在一定困难，并且求解精度不高。因此，我们选取启发式算法——粒子群算法对方程进行求解。

粒子群算法的介绍:

$$v_{i,d}(t+1) = w \cdot v_{i,d}(t) + c_1 \cdot r_1 \cdot (pbest_{i,d} - x_{i,d}(t)) + c_2 \cdot r_2 \cdot (gbest_d - x_{i,d}(t)) \quad (2)$$

这个公式是粒子群算法中用于更新粒子速度的核心公式。它包含三个部分:

1. 惯性项 (Inertia): $w \cdot v_{i,d}(t)$
   这部分表示粒子当前的速度对下一时刻速度的影响。w 是惯性权重，控制前一速度的影响程度。
2. 个体认知项 (Cognitive): $c_1 \cdot r_1 \cdot (pbest_{i,d} - x_{i,d}(t))$
   这部分表示粒子对自己历史最优位置的认知。
   c_1 是个体认知加速常数，控制这部分的影响。
   r_1 是一个随机数，增加搜索的随机性。
   $pbest_{i,d}$ 是粒子 i 在维度 d 上的个体最优位置。
   $x_{i,d}(t)$ 是粒子 i 在时间 t 在维度 d 上的当前位置。
3. 社会经验项 (Social): $c_2 \cdot r_2 \cdot (gbest_d - x_{i,d}(t))$
   这部分表示粒子对群体最优位置的认知。
   c_2 是社会经验加速常数，控制这部分的影响。
   r_2 是另一个随机数。
   $gbest_d$ 是全局最优在维度 d 上的位置。
   通过这三部分的综合作用, 粒子的速度会不断更新, 从而引导粒子向全局最优位置移动。这就是粒子群算法的核心思想。

粒子位置的更新过程,其中 $x_{i,d}(t)$ 是粒子 i 在维度 d 上在时间 t 的当前位置,$v_{i,d}(t+1)$ 是粒子 i 在维度 d 上在时间 $t+1$ 的速度，$x_{i,d}(t+1)$ 是粒子 i 在维度 d 上在时间 $t+1$ 的更新后的位置。

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \quad (3)$$

我们利用粒子群算法，针对发射信号为编号 FY00、FY0a、FY0b，接收信号为编号 FY0k 无人机，角度为 α、β、γ，定义如下适应度函数：

$$\sum_{\theta(i,j)=\alpha(0,a),\beta(0,b),\gamma(a,b)} D_{ik}^2 + D_{jk}^2 - D_{ij}^2 - 2cos\theta \cdot D_{ik}D_{jk} \quad (4)$$

从而通过粒子群算法求解 FY0k 对应坐标。

- **模型评估**

为验证模型准确性，对模型计算结果进行评估：
首先，建立极坐标仿真模型，随机挑选圆周上两架无人机作为发射信号的无人机，即 B、C，再在剩余无人机中挑选一架作为接收信号无人机，即 A，通过引入随机数函数给予 A 一个极坐标下的半径与角度的较小偏移量（角度变化范围 [-2,2]，半径变化范围 [-15,15]），从而获得初始坐标体系

表 **1**　偏移后极坐标（示例）

无人机编号	极坐标 (r,θ)
FY01	(100,0)
FY02	(100,40)
FY03	(100,80)
FY04	(100,120)
FY05	(112.4328,160.3703)
FY06	(100,200)
FY07	(100,240)
FY08	(100,280)
FY09	(100,320)

计算出 α 、β 、γ 代入建立的模型求解 A 的坐标，共计进行 1000 次模拟试验。在 1000 次试验中，误差在可以忽略的范围内的次数为 999 次，正确率达 0.999，认为模型效果良好。

5.1.2 第二小问模型的建立与求解

- **模型改良**

在这一问中，我们需要运用“各无人机相对位置不变”这一规则来进一步获取信息，在第一小问的模型下，因为编号未知，我们要确定是由哪个编号的无人机发出的信号，所以增加一个正确位置下各无人机发射信号与接收信号得出的角度矩阵，将偏离队列无人机所接受到的角度信息，与既有角度信息进行匹配，即分别将正确角度与得到的实际角度相减，因为偏移量较小，角度影响不大，所以将差值绝对值相加，总和最小的角度组代表的编号即为第三架无人机的编号

表 **2**　示例：接收信号无人机：**FY06**

第三架无人机编号 FY0i	00_01	01_0i	00_0i
FY02	10	20	10
FY03	10	40	30
FY04	10	60	50
FY05	10	80	70
FY06	10	NaN	NaN
FY07	10	60	70
FY08	10	40	50
FY09	10	20	30
接收真实角度	9.9061	39.3485	49.2546

由示例知，第三架无人机为 FY08，很明显可以看出，匹配成功度非常高，模型合理，于是我们对代码进行了修改，具体代码详见附件 B。

- **模型评估**

仍然通过仿真模型模拟实验，检测模型可靠性共计进行 5000 次模拟试验。在 5000 次随机试验中，误差在可以忽略的范围内的次数为 4859 次，正确率达 0.9718，认为模型效果良好。

得出结论，为实现无人机的有效定位，至少还需要一架无人机发射信号。

5.1.3 第三小问模型的建立与求解

首先，通过问题分析，我们确定第一步的调整方案，我们选择 FY02、FY05、FY07 作为先调整的无人机，每次选 FY02、FY05、FY07 中的一个接收信号，其余与 FY00、FY01 一起发射信号，结合角度矩阵判断角度来源无人机编号 (如示例中可明晰 a 为 0，b 为 1，c 为 2，d 为 7)

表 **3**　角度矩阵部分示例（以接收信号为无人机 **FY05** 为例）

发射编号 0a	发射编号 0b	发射编号 0c	发射编号 0d	接收编号 0e	0a_0b 度数	0a_0c 度数	0a_0d 度数	0b_0c 度数	0b_0d 度数	0c_0d 度数
0	1	2	7	5	10	30	50	20	60	80
0	1	7	2	5	10	50	30	60	20	80
0	2	1	7	5	30	10	50	20	80	60
0	2	7	1	5	30	50	10	80	20	60
1	0	2	7	5	10	20	60	30	50	80
1	0	7	2	5	10	60	20	50	30	80
真实 0a	真实 0b	真实 0c	真实 0d	5	10.1728	30.12	52.2395	19.9472	62.4123	82.3595

将接收到的四个角度组合成三组，每组为仅涉及三个编号的三个角度，再结合第一问模型求解相对正确位置

表 **4**　发射与接收信号次序

接收信号	发射信号组	分组运算
FY02	FY00、FY01、FY05、FY07	FY00　FY05　FY07
		FY00　FY01　FY07
		FY00　FY01　FY05
FY05	FY00、FY01、FY02、FY07	FY00　FY02　FY07
		FY00　FY01　FY02
		FY00　FY01　FY07
FY07	FY00、FY01、FY02、FY05	FY00　FY02　FY05
		FY00　FY01　FY02
		FY00　FY01　FY05

结合三次单次发射信号组与正确位置矩阵通过以正确位置代替发射信号无人机未知位置的粒子群算法得到的相对位置 p_1、p_2、p_3，为三个二维向量表示的位置坐标，再算出平均相对位置 p，通过

$$\Delta p = \frac{100}{|p|} \cdot p - p \tag{5}$$

算出相对移动距离并改变接收信号无人机位置，通过多次迭代优化，使 FY02、FY05、FY07 位置接近正确位置并且与 FY00 距离 100 米。

这时与正确位置的偏移非常小，接着再结合接收信号无人机正确位置作进一步粒子群优化，通过

$$\Delta p = p - R_i \tag{6}$$

算出相对移动距离并改变无人机位置，依然采用迭代优化，使得无人机抵达正确位置。

表 **5**　优化 **FY02**、**FY05**、**FY07** 后的极坐标（精确到 **0.01**）

无人机编号	极坐标长度 r	极坐标角度 θ
FY01	100.00	0.00
FY02	100.00	40.00
FY03	112.00	80.21
FY04	105.00	120.25
FY05	100.00	160.00
FY06	112.00	199.96
FY07	100.00	240.00
FY08	98.00	279.83
FY09	112.00	319.72

接着，在 FY00、FY01、FY02、FY05、FY07 编号已知、位置正确的情况下，结合问题一模型，遍历其余无人机位置，并利用公式（6）算出移动方向与距离，进而调整至正确位置。

表 **6**　最终位置（精确到 **0.00001**）

无人机编号	极坐标长度 r	极坐标角度 θ
FY01	100.00000	0.00000
FY02	100.00000	40.00000
FY03	100.00000	80.00000
FY04	100.00000	120.00000
FY05	100.00000	160.00000
FY06	100.00000	200.00000
FY07	100.00000	240.00000
FY08	100.00000	280.00000
FY09	100.00000	320.00000

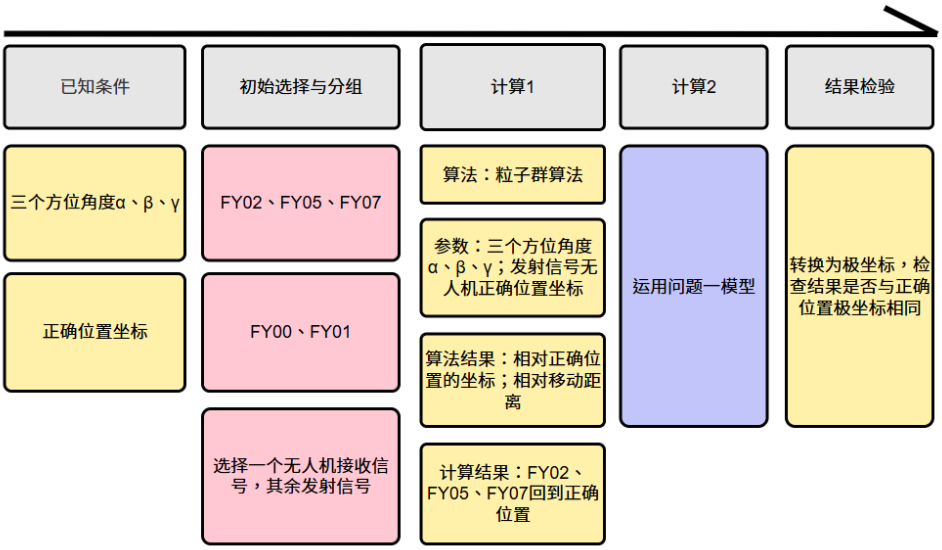


图 **6**　第三小问流程图

5.2 问题二

5.2.1 等边三角形角度修正模型的建立与求解

挑选位于等边三角形端点上的三架无人机（例如：FY01、FY02、FY03），如图，A、B、C 分别代表三架无人机。我们让其中两架发射信号，另外一架接收信号并且进行调整。

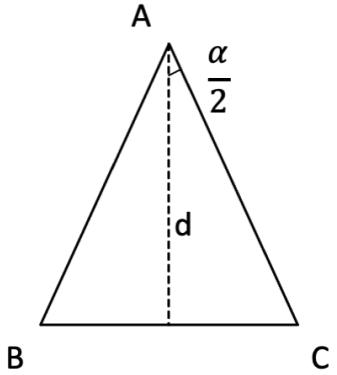


图 7　三架无人机示意图

由于偏移量在合理范围内，所以在 A 接收到 α 角时，假定 BC 间距离为正确位置之间的距离（如：50m）；并且假定 A 在 BC 的垂直平分线上，可以得到：

$$d = \frac{BC}{2tan\frac{\alpha}{2}} \tag{7}$$

此时，带入 B、C 的正确坐标，我们能得到 $d_0 = \frac{\sqrt{3}}{2}BC$，根据 B、C 的正确坐标以及 $\Delta d = d - d_0$, 我们可以对 A 的位置进行调整，让 A 接收到的角度更加接近 60 度。

在 A、B、C 中循环选择无人机来接收信号，对三架无人机之间的角度不断迭代优化，我们最终可以使得 A、B、C 三架无人机所接收的角度都为 60 度。

表 7　角度调整的仿真运行过程示例

无人机编号	偏移后初始极坐标		角度修正后极坐标		正确位置极坐标	
	极坐标长度 r	极坐标角度 θ	极坐标长度 r	极坐标角度 θ	极坐标长度 r	极坐标角度 θ
FY01	0	0	0	0	0	0
FY02	50.80388	-30.1231	50.43847	-30	50	-30
FY03	50.15306	29.66085	50.43847	30	50	30
FY04	99.04043	-29.8348	98.94773	-29.8543	100	-30
FY05	85.9863	-0.51429	85.87682	-0.49808	86.60254	0
FY06	99.15197	29.94804	99.05939	30.00152	100	30
FY07	150.6893	-30.0459	150.5968	-30.053	150	-30
FY08	131.5654	-11.1233	131.4583	-11.117	132.2876	-10.8934
FY09	132.6979	10.56613	132.5906	10.59312	132.2876	10.89339
FY10	151.1124	30.17081	151.0201	30.21188	150	30
FY11	200.4308	-29.7101	200.338	-29.7111	200	-30
FY12	181.2042	-16.3023	181.0998	-16.2966	180.2776	-16.1021
FY13	173.5089	-0.1821	173.3994	-0.16527	173.2051	0
FY14	181.1455	16.19192	181.0411	16.22005	180.2776	16.10211
FY15	199.5094	30.29989	199.4172	30.33518	200	30

5.2.2 等边三角形边长计算模型的建立与求解

在调整完角度之后，无人机 A、B、C 此时位于一个等边三角形的端点上，此时，我们让其中一架无人机 C 沿某个信号来源的反向延长线方向移动一定距离 CD，

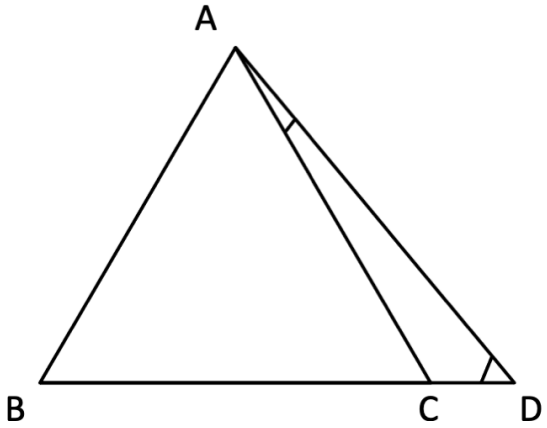


图 **8**　无人机短距离移动示意图

由三角关系和正弦定理，我们可以求出 A、C 间的距离，即等边三角形的距离。

$$\begin{cases} \angle ADC + \angle CAD + \angle ACD = 180° \\ \angle ACD = 120° \\ \frac{AC}{sin\angle ADC} = \frac{CD}{sin\angle CAD} \end{cases} \tag{8}$$

在得到 AC 之后，此时我们能够确定 A、B、C 的实际位置，进而可以让 A、B、C 三架无人机调整至正确位置。

表 **8**　边长调整的仿真运行过程示例

无人机编号	角度修正后极坐标		边长修正后极坐标		正确位置极坐标	
	极坐标长度 r	极坐标角度 θ	极坐标长度 r	极坐标角度 θ	极坐标长度 r	极坐标角度 θ
FY01	0	0	0	0	0	0
FY02	50.43847	-30	50.00002	-30	50	-30
FY03	50.43847	30	50.00002	30	50	30
FY04	98.94773	-29.8543	98.94773	-29.8543	100	-30
FY05	85.87682	-0.49808	85.87682	-0.49808	86.60254	0
FY06	99.05939	30.00152	99.05939	30.00152	100	30
FY07	150.5968	-30.053	150.5968	-30.053	150	-30
FY08	131.4583	-11.117	131.4583	-11.117	132.2876	-10.8934
FY09	132.5906	10.59312	132.5906	10.59312	132.2876	10.89339
FY10	151.0201	30.21188	151.0201	30.21188	150	30
FY11	200.338	-29.7111	200.338	-29.7111	200	-30
FY12	181.0998	-16.2966	181.0998	-16.2966	180.2776	-16.1021
FY13	173.3994	-0.16527	173.3994	-0.16527	173.2051	0
FY14	181.0411	16.22005	181.0411	16.22005	180.2776	16.10211
FY15	199.4172	30.33518	199.4172	30.33518	200	30

5.2.3 锥形编队定位模型的建立与求解

在三架无人机已经调整至正确位置的基础上，利用问题一第一小问的模型对剩下的无人机进行定位调整。

由于参与定位的无人机不能与发射信号的任意两架无人机在同一直线上，我们这里分两次定位调整。

第一次由 FY01、FY02、FY03 发射信号，由 FY05、FY08、FY09、FY12、FY13、FY14 接收信号，进行调整。

第二次则由 FY01、FY05、FY13 发射信号，由 FY04、FY06、FY07、FY10、FY11、FY15 接收信号，进行调整。最终我们能将所有无人机的位置调整至正确位置上。

表 **9**　仿真示例最终修正结果

无人机编号	边长修正后极坐标		修正其余无人机最终极坐标		正确位置极坐标	
	极坐标长度 r	极坐标角度 θ	极坐标长度 r	极坐标角度 θ	极坐标长度 r	极坐标角度 θ
FY01	0	0	0	0	0	0
FY02	50.00002	-30	50.00002	-30	50	-30
FY03	50.00002	30	50.00002	30	50	30
FY04	98.94773	-29.8543	100	-30	100	-30
FY05	85.87682	-0.49808	86.60258	-7.6279E-07	86.60254	0
FY06	99.05939	30.00152	100	30	100	30
FY07	150.5968	-30.053	150.0001	-30	150	-30
FY08	131.4583	-11.117	132.2876	-10.8934	132.2876	-10.8934
FY09	132.5906	10.59312	132.2876	10.89339	132.2876	10.89339
FY10	151.0201	30.21188	150.0001	30	150	30
FY11	200.338	-29.7111	200.0001	-30	200	-30
FY12	181.0998	-16.2966	180.2776	-16.1021	180.2776	-16.1021
FY13	173.3994	-0.16527	173.2052	-3.8892E-07	173.2051	0
FY14	181.0411	16.22005	180.2776	16.10211	180.2776	16.10211
FY15	199.4172	30.33518	200.0001	30	200	30

5.2.4 模型评估

通过仿真模型模拟实验，检测模型可靠性共计进行 1000 次模拟试验。在 1000 次随机试验中，误差在可以忽略的范围内的次数达 996 次，正确率达 0.996，认为模型效果良好。

六、 模型的评价

6.1 模型的优点

(1) 粒子群算法

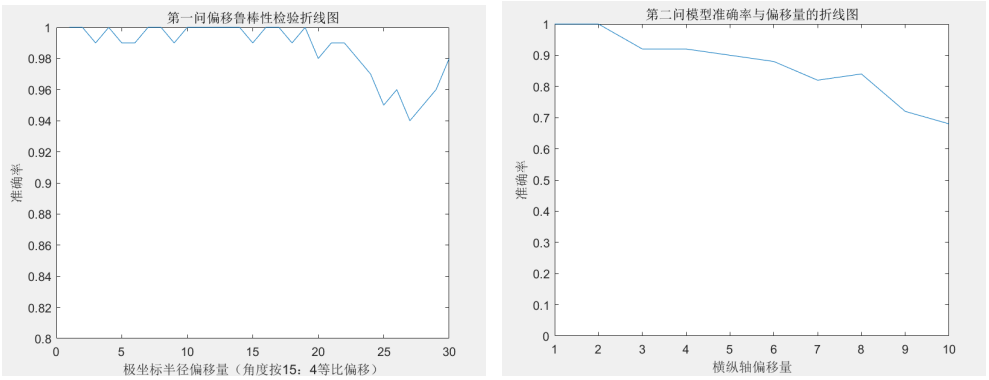
简单易实现：粒子群算法易于理解和实现，不需要太多高级数学知识。

全局搜索能力：PSO 能够在大范围内搜索解空间，并且具有一定的全局搜索能力。

容易并行化：粒子之间的独立性使得 PSO 易于并行化，可以有效地利用多核计算资源

(2) 鲁棒性好

我们通过更改仿真模型的偏移参数并计算准确率，在偏移幅度较大的情况下仍能保持较好的鲁棒性



(a) 第一问模型鲁棒性（半径 100m）　　(b) 第二问模型鲁棒性（相距 50m）

图 **9**　鲁棒性研究

(3) 准确率高

我们在研究过程中通过大量仿真实验，施加随机偏移量，所有模型的准确性仍维持在 0.95 以上，模型定位及调整结果的准确率非常高。

6.2 模型的缺点

(1) 粒子群算法

过早收敛：粒子群算法容易陷入局部最优解，特别是在高维空间和复杂的优化问题中。

参数敏感：PSO 的性能很大程度上取决于参数的选择，不同问题需要调整不同的参数。

缺乏理论支持：PSO 缺乏深入的理论基础，对于其工作原理的解释并不十分清晰。

参考文献

[1] 杨维, 李歧强. 粒子群优化算法综述 [J]. 中国工程科学,2004,6(5):87-94. DOI:10.3969/j.issn.1009-1742.2004.05.018.

[2] 李爱国, 覃征, 鲍复民, 等. 粒子群优化算法 [J]. 计算机工程与应用,2002,38(21):1-3,17. DOI:10.3321/j.issn:1002-8331.2002.21.001.

[3] 吕振肃, 侯志荣. 自适应变异的粒子群优化算法 [J]. 电子学报,2004,32(3):416-420. DOI:10.3321/j.issn:0372-2112.2004.03.016.

[4] 徐敬. 舰艇对海上目标纯方位无源定位研究 [D]. 大连理工大学,2002.

[5] 孙洪胜. 基于方位角测量的无源定位算法研究 [D]. 哈尔滨工程大学,2010.

[6] 辛沙欧. 纯角度量测下多无人机协同目标跟踪技术研究 [D]. 南京理工大学,2021.DOI:10.27241/d.cnki.gnjgu.2021.002226.

附录 A　第一小问粒子群算法

```
dev_sum=zeros(1,1000);
for j=1:1000
% 计算每个飞机的正确极坐标
right_position = zeros(9, 2);
for i = 1:9
right_position(i, :) = [100, (i - 1) * 40];
end
disp("正确位置为:")
disp(right_position)
% 假设飞机偏移
indices = randperm(9, 3); % 生成随机排列的索引，并取3个
right_num1 = indices(1);
right_num2 = indices(2);
wrong_num3 = indices(3);
% 设定偏移坐标
r = rand(1)*2-1;
a = right_position(wrong_num3, 1);
b = right_position(wrong_num3, 2);
right_position(wrong_num3, 2) = b + r;
r = rand()*30-15;
right_position(wrong_num3, 1) = a + r;

% 极坐标转化为直角坐标
for i = 1:9
a = right_position(i, 1);
b = right_position(i, 2);
right_position(i, :) = [a * cos(b / 360 * 2 * pi), a * sin(b / 360 * 2 * pi)];
end
%disp(right_position)

fy00_position = [0, 0];

% 计算夹角
A = fy00_position;
B = right_position(right_num1, :);
C = right_position(right_num2, :);
P = right_position(wrong_num3, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
```

```
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);

initial_guess = [100*cos((wrong_num3-1)/9*2*pi), 100*sin((wrong_num3-1)/9*2*pi)]; % 初始猜测值
disp("初始解为: ")
disp(initial_guess)
% 定义适应度函数
fitnessFunction = @(vars) sum([
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BA^2 -
    2*cos1*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - B(1))^2 + (vars(2) -
    B(2))^2);
(vars(1) - C(1))^2 + (vars(2) - C(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BC^2 -
    2*cos2*sqrt((vars(1) - C(1))^2 + (vars(2) - C(2))^2)*sqrt((vars(1) - B(1))^2 + (vars(2) -
    B(2))^2);
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - C(1))^2 + (vars(2) - C(2))^2 - CA^2 -
    2*cos3*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - C(1))^2 + (vars(2) -
    C(2))^2)
].^2);

% 使用粒子群优化求解
opts = optimoptions('particleswarm', 'Display', 'iter', 'InitialSwarmMatrix',initial_guess);
[x, fval] = particleswarm(fitnessFunction, 2, [], [], opts);
dev=(x(1)-right_position(wrong_num3,1))^2+(x(2)-right_position(wrong_num3,2))^2;
dev_sum(j)=dev;
% 显示解
disp(['x 的解: ', num2str(x(1))])
disp(['y 的解: ', num2str(x(2))])
end
percent=sum(dev_sum<1)/1000;
disp(["准确率:",percent]);
```

附录 B　第二小问算法代码

```
dev_sum=zeros(1,5000);
for j=1:5000
% 计算每个飞机的正确极坐标
right_position = zeros(9, 2);
for i = 1:9
right_position(i, :) = [100, (i - 1) * 40];
end
%正确坐标的直角坐标系
right_position_temp=zeros(9,2);
for i = 1:9
a = right_position(i, 1);
```

```
b = right_position(i, 2);
right_position_temp(i, :) = [a * cos(b / 360 * 2 * pi), a * sin(b / 360 * 2 * pi)];
end
% 假设飞机偏移
indices = randperm(9, 3); % 生成随机排列的索引，并取3个
right_num1 = 1;
k=1;
if indices(k)==1
k=k+1;
end
right_num2 = indices(k);
k=k+1;
if indices(k)==1
k=k+1;
end
wrong_num3 = indices(k);
%计算正确坐标各角度情况
true_degree=zeros(8,3);
for i=2:9
A = fy00_position;
B = right_position_temp(right_num1, :);
C = right_position_temp(i, :);
P = right_position_temp(wrong_num3, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);
true_degree(i-1,:)=[acos(cos1)*180/pi,acos(cos2)*180/pi,acos(cos3)*180/pi];
end
% 设定偏移坐标
r = rand(1)*2-1;
a = right_position(wrong_num3, 1);
b = right_position(wrong_num3, 2);
right_position(wrong_num3, 2) = b + r;
r = rand()*30-15;
right_position(wrong_num3, 1) = a + r;

% 极坐标转化为直角坐标
for i = 1:9
a = right_position(i, 1);
b = right_position(i, 2);
right_position(i, :) = [a * cos(b / 360 * 2 * pi), a * sin(b / 360 * 2 * pi)];
```

```
end
%disp(right_position)

fy00_position = [0, 0];

% 计算夹角
A = fy00_position;
B = right_position(right_num1, :);
C = right_position(right_num2, :);
P = right_position(wrong_num3, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);
%挑选最合适的飞机
true_degree=true_degree-[acos(cos1)*180/pi,acos(cos2)*180/pi,acos(cos3)*180/pi];
minSum = Inf;
minIndex = 0;
for i = 1:8
if i+1==wrong_num3
continue;
end
currentSum = true_degree(i, 2)^2 + true_degree(i, 3)^2;
if currentSum < minSum
minSum = currentSum;
minIndex = i;
end
end
C=right_position(minIndex+1,:);
% 生成初始解矩阵
num_particles = 50; % 粒子数
a=100-12+24*rand(num_particles, 1);
b=(wrong_num3-1)*40-0.5 + rand(num_particles, 1);
initial_guess = [a .* (cos(b/360*2*pi)),a .* (sin(b/360*2*pi))];

% 设置粒子群优化的参数
opts = optimoptions('particleswarm', ...
'Display', 'iter', ...   % 显示迭代过程
'MaxIterations', 500, ... % 最大迭代次数
'SwarmSize', num_particles, ... % 粒子数
'InitialSwarmMatrix', initial_guess ... % 初始解
);
```

```
% 定义适应度函数
fitnessFunction = @(vars) sum([
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BA^2 -
    2*cos1*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - B(1))^2 + (vars(2) -
    B(2))^2);
(vars(1) - C(1))^2 + (vars(2) - C(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BC^2 -
    2*cos2*sqrt((vars(1) - C(1))^2 + (vars(2) - C(2))^2)*sqrt((vars(1) - B(1))^2 + (vars(2) -
    B(2))^2);
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - C(1))^2 + (vars(2) - C(2))^2 - CA^2 -
    2*cos3*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - C(1))^2 + (vars(2) -
    C(2))^2)
].^2);

% 使用粒子群优化求解
[x, fval] = particleswarm(fitnessFunction, 2, [], [], opts);
dev=(x(1)-right_position(wrong_num3,1))^2+(x(2)-right_position(wrong_num3,2))^2;
dev_sum(j)=dev;
% 显示解
disp(['x 的解: ', num2str(x(1))])
disp(['y 的解: ', num2str(x(2))])
end
percent=sum(dev_sum<1)/5000;
disp(["准确率:",percent]);
```

附录 C　第三小问算法代码

```
    %先初始化生成
    %real_position 目标位置
    %right_position 目前位置
    tmp=initial();
    real_position=tmp(1:9,1:2);
    right_position=tmp(1:9,3:4);


    num=[2,5,7];
    %先优化距离，让大家都在一个圆上,优化10次
    for i=1:20
    for j=1:3
    k=num(mod(j,3)+1);
    pos1=cul_position(num(mod(j+2,3)+1),num(mod(j+1,3)+1),k,right_position,real_position);
    pos2=cul_position(1,num(mod(j+1,3)+1),k,right_position,real_position);
    pos3=cul_position(1,num(mod(j+2,3)+1),k,right_position,real_position);
    pos=pos1+pos2+pos3;
    pos=pos/3;
    pos=pos(1:2); %pos是算到的假坐标
```

```
delta_pos=(100/norm(pos))*pos-pos;
right_position(k,:)=delta_pos+right_position(k,:);
end
end

%优化10次
for s=1:20
for j=1:3
k=num(mod(j,3)+1);
pos1=cul_position(num(mod(j+2,3)+1),num(mod(j+1,3)+1),k,right_position,real_position);
pos2=cul_position(1,num(mod(j+1,3)+1),k,right_position,real_position);
pos3=cul_position(1,num(mod(j+2,3)+1),k,right_position,real_position);
pos=pos1+pos2+pos3;
pos=pos/3;
pos=pos(1:2); %pos是算到的假坐标
delta_pos=pos-real_position(k,:);
right_position(k,:)=right_position(k,:)-delta_pos;
end
end
values=[3,4,6,8,9];
right_num1=1;
right_num2=2;
for wrong_num3=values
pos=cul_position_2(right_num1,right_num2,wrong_num3,right_position,real_position);
delta_pos=pos-real_position(wrong_num3,:);
right_position(wrong_num3,:)=right_position(wrong_num3,:)-delta_pos;
end
%转换成极坐标
for i=1:9
right_position(i,:)=[norm(right_position(i,:)),atand(right_position(i,2)/right_position(i,1))];
end
```

```
function ini=initial()
% 计算每个飞机的正确极坐标
real_position = zeros(9, 2);
for i = 1:9
real_position(i, :) = [100, (i - 1) * 40];
end
% 极坐标转化为直角坐标
for i = 1:9
a = real_position(i, 1);
b = real_position(i, 2);
real_position(i, :) = [a * cos(b / 360 * 2 * pi), a * sin(b / 360 * 2 * pi)];
end

% 每个飞机初始极坐标
```

```
right_position ...
=[100,0;
98,40.10;
112,80.21;
105,119.75;
98,159.86;
112,199.96;
105,240.07;
98,280.17;
112,320.28];
disp("极坐标为:");
disp(right_position);


% 极坐标转化为直角坐标
for i = 1:9
a = right_position(i, 1);
b = right_position(i, 2);
right_position(i, :) = [a * cos(b / 360 * 2 * pi), a * sin(b / 360 * 2 * pi)];
end
disp("直角坐标为:");
disp(right_position);
ini=[real_position,right_position];
end
```

```
function position = cul_position(num1,num2,wrong_num,right_position,real_position)
%计算正确坐标各角度情况
% 初始化一个5*5*5的cell数组
true_degree = cell(5,5,5);
%
% % 填充这个cell数组，使得每个元素都是一个1*3的零向量
for i = 1:5
for j = 1:5
for k=1:5
true_degree{i, j, k} = zeros(1, 3); % 创建一个1*3的零向量并放入cell数组中的相应位置
end
end
end
num=[0,1,2,5,7];
for i=1:5
for j=1:5
for k=1:5
if i==j ||j==k||i==k||num(i)==wrong_num||num(j)==wrong_num||num(k)==wrong_num
continue;
end
if num(i)==0
```

```
A=[0,0];
else
A=real_position(num(i), :);
end
if num(j)==0
B=[0,0];
else
B=real_position(num(j), :);
end
if num(k)==0
C=[0,0];
else
C=real_position(num(k), :);
end
P = real_position(wrong_num, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);
true_degree{i,j,k}=[acos(cos1)*180/pi,acos(cos2)*180/pi,acos(cos3)*180/pi];
end
end
end
% 计算夹角
A = [0,0];
B = right_position(num1, :);
C = right_position(num2, :);
P = right_position(wrong_num, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);
%挑选最合适的飞机
for i=1:5
for j=1:5
for k=1:5
true_degree{i,j,k}=true_degree{i,j,k}-[acos(cos1)*180/pi,acos(cos2)*180/pi,acos(cos3)*180/pi];
```

```
end
end
end
minSum = Inf;
minIndex_1 = 0;
minIndex_2 = 0;
minIndex_3=0;
for i = 1:5
for j = 1:5
for k=1:5
if i==j||i==k||j==k
continue
end
if num(i)==wrong_num || num(j)==wrong_num||num(k)==wrong_num
continue
end
currentSum = true_degree{i,j,k}(1)^2 + true_degree{i, j,k}(2)^2 + true_degree{i,
    j,k}(3)^2;
if currentSum < minSum
minSum = currentSum;
minIndex_1 = num(i);
minIndex_2=num(j);
minIndex_3=num(k);
end
end
end
end
%带入理想坐标
if minIndex_1==0
A=[0,0];
else
A=real_position(minIndex_1,:);
end
if minIndex_2==0
B=[0,0];
else
B=real_position(minIndex_2,:);
end
if minIndex_3==0
C=[0,0];
else
C=real_position(minIndex_3,:);
end
% 生成初始解矩阵
num_particles = 50; % 粒子数
a=100-12+24*rand(num_particles, 1);
b=(wrong_num-1)*40-0.5 + rand(num_particles, 1);
```

```
initial_guess = [a .* (cos(b/360*2*pi)),a .* (sin(b/360*2*pi))];

% 设置粒子群优化的参数
opts = optimoptions('particleswarm', ...
'Display', 'iter', ...   % 显示迭代过程
'MaxIterations', 500, ... % 最大迭代次数
'SwarmSize', num_particles, ... % 粒子数
'InitialSwarmMatrix', initial_guess ... % 初始解
);
%带入理想坐标
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
% 定义适应度函数
fitnessFunction = @(vars) sum([
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BA^2
    - 2*cos1*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - B(1))^2 +
    (vars(2) - B(2))^2);
(vars(1) - C(1))^2 + (vars(2) - C(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BC^2
    - 2*cos2*sqrt((vars(1) - C(1))^2 + (vars(2) - C(2))^2)*sqrt((vars(1) - B(1))^2 +
    (vars(2) - B(2))^2);
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - C(1))^2 + (vars(2) - C(2))^2 - CA^2
    - 2*cos3*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - C(1))^2 +
    (vars(2) - C(2))^2)
].^2);

% 使用粒子群优化求解
[x, fval] = particleswarm(fitnessFunction, 2, [], [], opts);
position = [x, fval];
end
```

```
function position =
    cul_position_2(right_num1,right_num2,wrong_num3,right_position,real_position)
%CUL_POSITION_2 此处显示有关此函数的摘要
%   此处显示详细说明
%计算正确角度集合
% 初始化一个4*4*4的cell数组
true_degree = cell(3,3,3);
%
% % 填充这个cell数组，使得每个元素都是一个3x1的零向量
for i = 1:3
for j = 1:3
for k=1:3
true_degree{i, j, k} = zeros(1, 3); % 创建一个1x3的零向量并放入cell数组中的相应位置
end
end
```

```
end
num=[0,1,2];
for i=1:3
for j=1:3
for k=1:3
if i==j ||j==k||i==k
continue;
end
if num(i)==0
A=[0,0];
else
A=real_position(num(i), :);
end
if num(j)==0
B=[0,0];
else
B=real_position(num(j), :);
end
if num(k)==0
C=[0,0];
else
C=real_position(num(k), :);
end
P = real_position(wrong_num3, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);
true_degree{i,j,k}=[acos(cos1)*180/pi,acos(cos2)*180/pi,acos(cos3)*180/pi];
end
end
end
% 计算夹角
A = [0,0];
B = right_position(right_num1, :);
C = right_position(right_num2, :);
P = right_position(wrong_num3, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
```

```
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);
%挑选最合适的飞机
for i=1:3
for j=1:3
for k=1:3
true_degree{i,j,k}=true_degree{i,j,k}-[acos(cos1)*180/pi,acos(cos2)*180/pi,acos(cos3)*180/pi];
end
end
end
minSum = Inf;
minIndex_1 = 0;
minIndex_2 = 0;
minIndex_3=0;
for i = 1:3
for j = 1:3
for k=1:3
if i==j||i==k||j==k
continue
end
currentSum = true_degree{i,j,k}(1)^2 + true_degree{i, j,k}(2)^2 + true_degree{i,
    j,k}(3)^2;
if currentSum < minSum
minSum = currentSum;
minIndex_1 = num(i);
minIndex_2=num(j);
minIndex_3=num(k);
end
end
end
end
if minIndex_1==0
A=[0,0];
else
A=real_position(minIndex_1,:);
end
if minIndex_2==0
B=[0,0];
else
B=real_position(minIndex_2,:);
end
if minIndex_3==0
C=[0,0];
else
C=real_position(minIndex_3,:);
```

```
    end
    % 生成初始解矩阵
    num_particles = 50; % 粒子数
    a=100-12+24*rand(num_particles, 1);
    b=(wrong_num3-1)*40-0.5 + rand(num_particles, 1);
    initial_guess = [a .* (cos(b/360*2*pi)),a .* (sin(b/360*2*pi))];

    % 设置粒子群优化的参数
    opts = optimoptions('particleswarm', ...
    'Display', 'iter', ...   % 显示迭代过程
    'MaxIterations', 500, ... % 最大迭代次数
    'SwarmSize', num_particles, ... % 粒子数
    'InitialSwarmMatrix', initial_guess ... % 初始解
    );
    % 定义适应度函数
    fitnessFunction = @(vars) sum([
    (vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BA^2
        - 2*cos1*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - B(1))^2 +
        (vars(2) - B(2))^2);
    (vars(1) - C(1))^2 + (vars(2) - C(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BC^2
        - 2*cos2*sqrt((vars(1) - C(1))^2 + (vars(2) - C(2))^2)*sqrt((vars(1) - B(1))^2 +
        (vars(2) - B(2))^2);
    (vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - C(1))^2 + (vars(2) - C(2))^2 - CA^2
        - 2*cos3*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - C(1))^2 +
        (vars(2) - C(2))^2)
    ].^2);

    % 使用粒子群优化求解
    [x, ~] = particleswarm(fitnessFunction, 2, [], [], opts);
    position=x;
    end
```

附录 D　问题二算法代码

```
position=initial();
right_position=position(:,1:2); %正确位置
real_position=position(:,3:4); %目前位置
%real_position=real_position-real_position(5,:);
%right_position=right_position-right_position(5,:);
z_1=[-1,0];
z_2=[0.5,-0.5*sqrt(3)];
z_3=[0.5,0.5*sqrt(3)];
while abs(cal_degree(1,2,3,right_position,real_position)-60/180*pi)>0.00000001 ||
    abs(cal_degree(3,2,1,right_position,real_position)-60/180*pi)>0.00000001 ||
    abs(cal_degree(1,3,2,right_position,real_position)-60/180*pi)>0.00000001
```

```
delta_distance_3=delta_distance(1,2,3,right_position,real_position);
real_position(3,:)=real_position(3,:)+delta_distance_3*z_3;
delta_distance_1=delta_distance(3,2,1,right_position,real_position);
real_position(1,:)=real_position(1,:)+delta_distance_1*z_1;
delta_distance_2=delta_distance(1,3,2,right_position,real_position);
real_position(2,:)=real_position(2,:)+delta_distance_2*z_2;
end
real_position=real_position-real_position(1,:);
%变更坐标系
%转换成极坐标
for i=2:15
real_position(i,:)=[norm(real_position(i,:)),atand(real_position(i,2)/real_position(i,1))];
end
temp=real_position(2,2)+30;
for i=2:15
real_position(i,:)=real_position(i,:)-temp;
end
real_position(1,:)=[0,0];
for i=2:15
real_position(i,:)=[real_position(i,1)*cos(real_position(i,2)/180*pi),real_position(i,1)*sin(real_position(i
end
x=bianchang(2,1,3,real_position);
delta_distance_2=[50*sqrt(3)/2-x*sqrt(3)/2,x/2-50/2];
real_position(2,:)=real_position(2,:)+delta_distance_2;
delta_distance_3=[50*sqrt(3)/2-x*sqrt(3)/2,50/2-x/2];
real_position(3,:)=real_position(3,:)+delta_distance_3;
%更换正确坐标系朝向
for i=2:15
right_position(i,:)=[norm(right_position(i,:)),atand(right_position(i,2)/right_position(i,1))];
end
temp=right_position(2,2)+30;
right_position(1,:)=[0,0];
for i=2:15
right_position(i,:)=[right_position(i,1)*cos(right_position(i,2)/180*pi),right_position(i,1)*sin(right_positi
end
%已知01，02，03正确坐标之后调整其他坐标
for i=[5,8,9,12,13,14]
pos=cul_position(1,2,3,i,right_position,real_position);
real_position(i,:)=real_position(i,:)-(pos(1:2)-right_position(i,:));
end
for i=[4,6,7,10,11,15]
pos=cul_position(1,5,13,i,right_position,real_position);
real_position(i,:)=real_position(i,:)-(pos(1:2)-right_position(i,:));
end
```

```
%初始化矩阵
```

```
function position=initial()
%三角形边长
a=50;
%xy坐标误差范围在正负delta以内
delta=1;
% 向量表示
A=[-sqrt(3)*a/2,a/2];%表示斜向左上方的单位向量
B=[-sqrt(3)/2*a,-a/2];%表示斜向左下方的单位向量
C=[0,-a];%表示向下的向量

position=zeros(15,4);
%以fy01为原点做直角坐标系
num=1;
for i=0:4
for j=0:i
position(num,1:2)=i*A + j*C;
num=num+1;
end
end
position(:,3:4)=position(:,1:2);
for i=2:15
position(i,3)=position(i,3)+delta*2*rand()-delta;
position(i,4)=position(i,4)+delta*2*rand()-delta;
end
end
```

```
function delta_pos=delta_distance(num1,num2,k,right_position,real_position)
% 边长
a=50;
% 计算夹角
af=cal_degree(num1,num2,k,right_position,real_position);
%计算相对移动距离
delta_pos=(25/tan(af/2)-25*sqrt(3))*0.000001;
end
```

```
function af = cal_degree(num1,num2,k,right_position,real_position)
%DEGREE 此处显示有关此函数的摘要
%   此处显示详细说明
A = real_position(k, :);
B = real_position(num1, :);
C = real_position(num2, :);
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1=(CA^2+BA^2-BC^2)/(2*CA*BA);
af=acos(cos1);
```

```
end
```

```
function x=bianchang(num1,num2,num3,real_position)
% 这里的num1最好是右下角的飞机
delta_y=1;
real_position(num1,2)=real_position(num1,2)-delta_y;

% 计算夹角
A = real_position(num1, :);
B = real_position(num2, :);
C = real_position(num3, :);
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1=(CA^2+BA^2-BC^2)/(2*BA*CA);
af=acos(cos1);
af2=pi/3-af;
x=delta_y * sin(af) / sin(af2);
end
```

```
function position = cul_position(numA,num1,num2,wrong_num,right_position,real_position)
% 计算夹角
A = real_position(numA, :);
B = real_position(num1, :);
C = real_position(num2, :);
P = real_position(wrong_num, :);
PA = sqrt(sum((A - P).^2));
PB = sqrt(sum((B - P).^2));
PC = sqrt(sum((C - P).^2));
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
cos1 = (PA^2 + PB^2 - BA^2) / (2 * PA * PB);
cos2 = (PC^2 + PB^2 - BC^2) / (2 * PC * PB);
cos3 = (PA^2 + PC^2 - CA^2) / (2 * PA * PC);

% 生成初始解矩阵
num_particles = 50; % 粒子数
initial_guess = (rand(50, 2) - 0.5) * 2+right_position(wrong_num,:);

% 设置粒子群优化的参数
opts = optimoptions('particleswarm', ...
'Display', 'iter', ...   % 显示迭代过程
'MaxIterations', 500, ... % 最大迭代次数
'SwarmSize', num_particles, ... % 粒子数
'InitialSwarmMatrix', initial_guess ... % 初始解
```

```
);
%带入正确坐标
A = right_position(numA, :);
B = right_position(num1, :);
C = right_position(num2, :);
BA = sqrt(sum((A - B).^2));
CA = sqrt(sum((A - C).^2));
BC = sqrt(sum((B - C).^2));
% 定义适应度函数
fitnessFunction = @(vars) sum([
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BA^2
    - 2*cos1*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - B(1))^2 +
    (vars(2) - B(2))^2);
(vars(1) - C(1))^2 + (vars(2) - C(2))^2 + (vars(1) - B(1))^2 + (vars(2) - B(2))^2 - BC^2
    - 2*cos2*sqrt((vars(1) - C(1))^2 + (vars(2) - C(2))^2)*sqrt((vars(1) - B(1))^2 +
    (vars(2) - B(2))^2);
(vars(1) - A(1))^2 + (vars(2) - A(2))^2 + (vars(1) - C(1))^2 + (vars(2) - C(2))^2 - CA^2
    - 2*cos3*sqrt((vars(1) - A(1))^2 + (vars(2) - A(2))^2)*sqrt((vars(1) - C(1))^2 +
    (vars(2) - C(2))^2)
].^2);

lb=[right_position(wrong_num,1)-2,right_position(wrong_num,2)-2];
ub=[right_position(wrong_num,1)+2,right_position(wrong_num,2)+2];
% 使用粒子群优化求解
[x, fval] = particleswarm(fitnessFunction, 2, [],[], opts);
position = [x, fval];
end
```