基于投票机制及二分法的纯方位无源定位研究

摘要

本文研究的是集群无人机的纯方位无源定位方式，通过建立投票机制模型及二分移动路径法求解得到接收信号无人机的真实位置，从而对其进行调整。

整体问题从已知所有无偏差发射信号无人机的方向信息，到缺失一架发射信号无人机的编号信息，再到所有发射信号无人机均有偏差且编号未知，问题所给信息量逐渐减小。从少量信息中提取有效信息实现定位是本问题的关键所在。

针对问题一： 本题需要在已知 3 架无偏无人机给出的方向信息的情况下，给出接收端无人机的具体定位。将无人机抽象为四个点进行平面几何分析，发现随着接收端无人机相对其余三个点的位置发生改变，定位计算公式也会随之改变。因此我们用 $FY00$ 与其余两个发射源之间的连线将平面划分成**四个区域**，得到四个不同的计算公式。由于接收端无人机落在哪个区域初识是未知的，故可以通过方向信息计算出 4 个解。同时因为接收端无人机的位置偏移是略微的，故我们采用距离其理想位置最近的点为其真实位置。为检验模型的准确性，我们模拟了 100000 次随机的有偏移的初始位置，发现在**偏移误差小于 30%** 的情况下，定位的准确性均可达到 **100%**，说明模型效果良好。进一步扩大误差后，我们发现选取**圆心角为 40° 的两个发射源**相较其余三种 (80°,120°,160°) 定位错误率更低。

针对问题二： 题目要求利用若干编号未知的发射信号无人机实现定位。为了利用问题一中模型，考虑先确定发射信号无人机的编号，再调用问题一中模型求解。

首先我们尝试直接**线性划分**不同角度的信号、对应不同的信号来源无人机，建立了简单模型。由于精度不高，我们考虑待定位无人机在以理想位置为圆心的**误差圆**内随机游动，改变误差圆半径、计算不同半径下各可能信号源无人机形成的圆周角范围，再累积测得的真实角度落在各个信号源的角度范围的频次。通过上述**投票机制**，选择出最可能的信号源编号。接着我们代入此编号无人机求解对应的圆周角进行**角度验证**，若与测得的角度偏差超过限度，则认为编号确定错误。从而再引入一架发射信号的无人机，重复上述过程投票选出一对无人机编号组合，计算得到真实位置。

在模拟 10^6 次初始情况后，我们发现在 20% 的偏移误差下，投票机制模型能达到 100% 的正确性，而在 40% 的偏移误差下，仍有 **7%** 的错误率，而再引入一架无人机后，该错误率能降至 **2%**，符合有效定位的要求。

针对问题三： 本题需要将圆周上与理想位置有所偏差的 9 架无人机调整到尽可能接近理想位置。我们将 $FY01$ 设为基准无人机，使用**二分移动路径法**确定待测无人机和坐标原点的相对方向，再通过解三角形得到所有待测无人机到原点的距离，最后再用一轮指令调整至理想位置。对于题目给出的初始情况，我们使用 **253 次**发射信号或移动操作将所有无人机调整到理想位置附近，所有无人机的偏差距离和仅为 $0.000160328m$。我们还自行随机生成了 10^6 组不同的初始情况，根据我们的模型得出的方案进行调整，最终偏差距离和均**小于** $10^{-3}m$。

针对问题四： 本题将调整其他无人机编队队形中的无人机至理想位置，以题目中所给的锥形无人机编队图为例，我们考虑应用类似问题三的解决思路，我们将 $FY01$ 设为坐标原点，将 $FY05$ 设为基准无人机，使用类似的**二分移动路径法**和解三角形方法定位所有待测无人机位置，最后用一轮指令调整至理想位置。我们可以用 20 轮询问和调整将每个无人机的角度偏差降低到 10^{-5} 弧度之内。

关键词： 无源定位　误差圆　投票机制　二分移动路径法

1 问题的背景与重述

1.1 问题的背景

随着无人机制造技术的不断进展，无人机已经被广泛地运用到军事、物流、探测等多个领域，越来越多地进入国家生活的方方面面。而在与无人机相关的工程技术中，如何控制无人机机群保持一定的队形编队飞行，更是现在无人机技术大规模应用的重点与难点。在相关的控制方法研究中，利用无人机间相互信号收发定位的纯方位无源定位方法，以其抗干扰能力强、信号传输隐蔽、能很好适应不同的环境 [1]，而成为无人机编队飞行的一种良好解决方案。因此探究在纯方位无源定位方法下，无人机编队定位与队形调整的数学模型与算法实现具有重要的研究价值。

1.2 问题的重述

根据题目要求，本文旨在解决下列问题：

问题一：在圆形飞行编队中，利用圆心处的 FY00 和两架圆周上已知编号的无人机发射信号，尝试建立接收信号无人机的定位模型。

问题二：在圆形飞行编队中，利用 FY00、FY01 以及若干圆周上未知编号的无人机发射信号，建立接收信号无人机的定位模型。

问题三：仍在圆形飞行编队中，每次选择 FY00 和圆周上最多 3 架无人机遂行发射信号，依此确定调整方案使得多次调整后机群最终均匀分布在某个圆周上。

问题四：转为考虑在锥形飞行编队中，用类似问题三的方法确定信号发射方案与队形调整方案，使得多次调整后机群队形变为所要求的锥形队列。

2 问题分析

对于问题一：问题一要求在发射信号的无人机位置无偏差且编号已知的情况下，对被动接收信号的无人机进行定位。将 3 架已知无人机（A、B、C）与 1 架未知无人机（D）的位置抽象化，对其几何图形进行分析。根据 AB、AC 的连线将整个平面分成区域，D 点落在不同的区域即有不同的位置计算公式。由于在实际情况中，D 点所在的区域是未知的，4 个解均有可能，故我们选用离 D 点理想位置最近的点作为其真实位置。同时我们进一步探究了已知发射源的选取位置优劣性。

对于问题二：题目要求利用若干编号未知的发射信号无人机实现定位。首先我们利用不同信号源无人机与待定位无人机形成的圆周角不尽相同，直接对角度进行线性划分确定信号源。

进一步我们考虑待定位无人机位置的误差圆，对不同半径下的可能信号源进行统计，综合选出最可能的信号源编号。接着我们代入此编号无人机求解对应圆周角进行验证，若与测得的角度偏差超过限度，则认为编号确定错误。此时我们类似地再引入一架发射信号的无人机，进行下一轮投票，直至符合误差。

对于问题三：选取 $FY00$ 作为坐标原点，$FY01$ 作为基准无人机，通过二分法多次调整并发射信号测试 ，判断其他圆周上的无人机的方向，再通过解三角形得到其到坐标原点的距离。最后通过已确定位置的 $FY02$ 和相关角度信息计算 $FY01$ 的方向和距离，并一次调整令所有圆周上的无人机移动到理想位置附近。

对于问题四：选取 $FY01$ 作为坐标原点，$FY05$ 作为基准无人机，用和问题三类似的方法确定除了 $FY05$ 和 $FY13$ 的位置信息，最后通过已确定位置的 $FY02$ 和相关角度信息计算 $FY05$ 和 $FY13$ 的方向和距离，并全部调整到理想位置附近。

3 模型假设

1. 基于自身感知的高度信息，本文默认无人机均保持在同一个高度上飞行，即所有无人机在同一个平面上。
2. 假设第一问中的所有小问 9 架无人机理论上都应均匀分布在半径为 100m 的圆周上。
3. 假设无人机接收到的方向信息不存在误差。
4. 本题均拟采用纯方位无源定位的方法调整无人机的位置。
5. 假设当没有调整指令时，无人机编队中的每架飞机都在向相同方向以相同速度前进，即编队整体在向某个方向移动。
6. 假设无人机在调整位置时获得的是 (x,y) 型指令，即获得在 x 轴和 y 轴上的位置变化量。

4 符号说明

符号	含义	单位
O	FY00 编号无人机所在点	/
P	FY01 编号无人机的所在点	/
F	可能发射源	/
F_{truei}	真实的第 i 个发射源	/
J	接收点的真实位置	/
Q	误差圆编号（按半径依次编号）	/
I	误差圆的圆心编号	/
(O_{min}, O_{max})	包含 O 点的角度范围	°
(P_{min}, P_{max})	包含 O 点的角度范围	°
$vote_F$	发射源得票数	/
FYi	FY0i 编号无人机的理想位置	/
$FYi^{'}$	FY0i 编号无人机的真实位置	/
$FYi^{''}$	FY0i 编号无人机二分调整后的位置	/
$(\Delta X_{i,j}, \Delta Y_{i,j})$	第 i 次第 j 号无人机的移动指令	m

5 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 极坐标系的建立

为了更加明确地标注出各编号无人机的理想位置，我们选用包含角度信息的极坐标系来进行刻画。由于被动接收信号的无人机仅能接收到方向信息，即角度，故极坐标系相比直角坐标系更易表达清晰。

假设 9 架无人机 (编号 FY01~FY09) 均匀分布在一个半径为 100m 的圆周上。以编号 FY00 的无人机作为坐标原点，编号 FY00 与编号 FY01 之间的连线作为 X 轴，建立极坐标系如下图1。各个编号的理想位置坐标信息如下表：

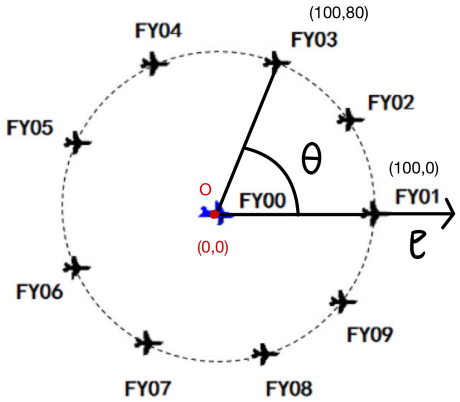


图 1: 极坐标系示意图

表 1：各编号无人机理想坐标

无人机编号	极坐标 (m,°)	无人机编号	极坐标 (m,°)	无人机编号	极坐标 (m,°)
FY00	(0,0)	FY01	(100,0)	FY02	(100,40)
FY03	(100,80)	FY04	(100,120)	FY05	(100,160)
FY06	(100,200)	FY07	(100,240)	FY08	(100,280)
FY09	(100,320)				

5.1.2 定位模型的建立

题目要求在已知未知定位无人机与任意两架发射源无人机间连线的夹角后，给出未知定位无人机当前的所在位置。由于发射源无人机的位置是无偏差的，故它们间的距离同样为已知信息。对该问题进行几何分析：

图中 A、B、C 三点为已知位置的发射源无人机（A 为原点，B、C 点在圆上按逆时针顺序排列），三点将平面区域划分为 4 各部分。D 点为需要求解的无人机的位置，可能落在任意一个区域中。如下图2所示：

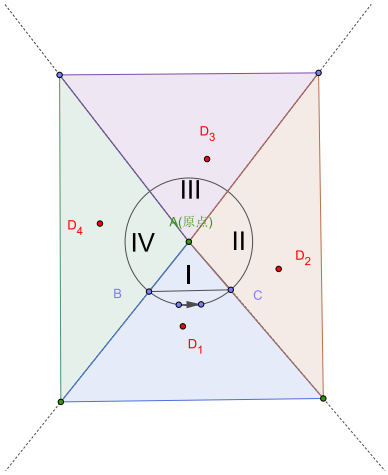


图 2: 区域划分示意图

其中，A、B 和 A、C 间的距离均为半径 r。已知的方向信息包含两个小角与小角组合形成的大角，实际有效信息仅含两个角度。为了方便计算，我们选择包含原点 A 的角度 α_1、α_2 以及发射源

无人机间的夹角 α_3 进行分析。

对每个部分的点分别建立定位模型如下：

利用正弦定理及角度关系对 ΔABD，ΔACD，ΔBCD 进行分析：

设 $\angle ABD=\theta_1$，$\angle ACD=\theta_2$，$\angle ADB=\alpha_1$，$\angle ADC=\alpha_2$，$\angle BAC=\alpha_3$，AD=l。

情况一：若 α_1、α_2 之间不存在包含关系，即 D 点落在 I、III 区域，分为两种情况讨论：

若 D 点落在 I 区域，如图3，则:

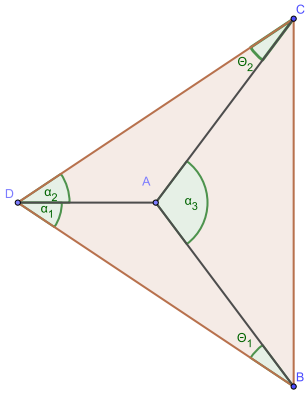


图 3: 区域 I 典型图

$$\begin{cases} \theta_1+\theta_2=\alpha_3-(\alpha_1+\alpha_2)=\theta_0 \\ \frac{sin\theta_1}{l}=\frac{sin\alpha_1}{r}=k_1 \\ \frac{sin\theta_2}{l}=\frac{sin\alpha_2}{r}=k_2 \end{cases} \tag{1}$$

$$\Rightarrow sin(\theta_0-\theta_1)=\frac{k_2}{k_1}sin\theta_1$$

$$\Rightarrow sin\theta_0\cdot cos\theta_1-cos\theta_0\cdot sin\theta_1=\frac{k_2}{k_1}sin\theta_1$$

$$\Rightarrow \theta_1=arctan(\frac{sin\theta_0}{\frac{sin\alpha_2}{sin\alpha_1}+cos\theta_0}),\theta_0=\alpha_3-(\alpha_1+\alpha_2)$$

在求解过程中，如果发现等式无法计算或出现正无穷的情况，则考虑 $\theta_1=90°$。

若 B 的极坐标为 (100,β)，则该种情况下，未知点的定位坐标为 (100×$\frac{sin\theta_1}{sin\alpha_1}$,$\beta$−($\pi$-$\theta_1$-$\alpha_1$))。

若 D 点落在 III 区域，如图4，则：

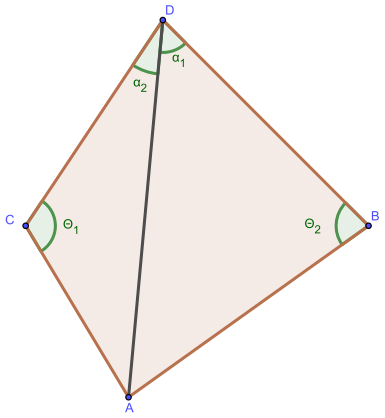


图 4: 区域 III 典型图

无论 D 点在 ΔABC 内部还是外部，所建立的方程保持不变。

$$\begin{cases} \theta_1 + \theta_2 = 2\pi - (\alpha_1 + \alpha_2 + \alpha_3) \\ \frac{sin\theta_1}{l} = \frac{sin\alpha_1}{r} = k_1 \\ \frac{sin\theta_2}{l} = \frac{sin\alpha_2}{r} = k_2 \end{cases} \tag{2}$$

$$\Rightarrow \theta_1 = arctan\frac{-sin(\alpha_1 + \alpha_2 + \alpha_3)}{\frac{sin\alpha_2}{sin\alpha_1} + cos(\alpha_1 + \alpha_2 + \alpha_3)}$$

或 $\theta_1 = 90°$。

若 B 的极坐标为 (100,β)，则该种情况下，未知点的定位坐标为 (100×$\frac{sin\theta_1}{sin\alpha_1}$,$\beta$+($\pi$-$\theta_1$-$\alpha_1$))。

情况二：若 α_1、α_2 之间存在包含关系，即 D 点落在 II、IV 区域，两种可能性如图56所示；

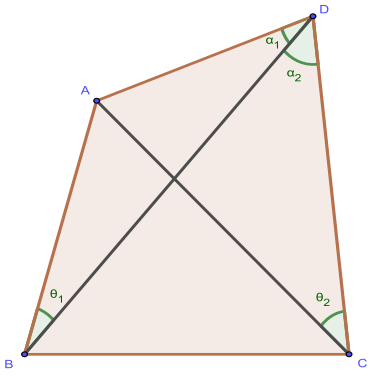


图 5: 区域 II 典型图

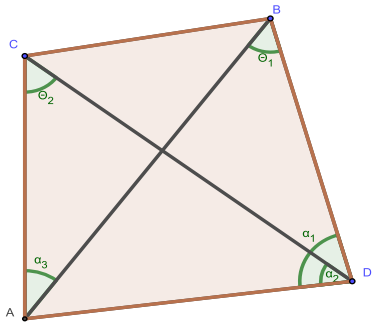


图 6: 区域 IV 典型图

当 $\alpha_1 < \alpha_2$ 时:

$$\begin{cases} \theta_1 - \theta_2 = \alpha_2 - (\alpha_1 + \alpha_3) = \theta_0 \\ \frac{sin\theta_1}{l} = \frac{sin\alpha_1}{r} = k_1 \\ \frac{sin\theta_2}{l} = \frac{sin\alpha_2}{r} = k_2 \end{cases} \tag{3}$$

$$\Rightarrow \theta_1 = arctan\frac{sin\theta_0}{cos\theta_0 - \frac{sin\alpha_2}{sin\alpha_1}}, \theta_0 = \alpha_2 - (\alpha_1 + \alpha_3)$$

或 $\theta_1 = 90°$。

若 B 的极坐标为 (100,β)，则该种情况下，未知点的定位坐标为 (100×$\frac{sin\theta_1}{sin\alpha_1}$,$\beta$+($\pi$-$\theta_1$-$\alpha_1$))。

当 $\alpha_1 > \alpha_2$ 时:

$$\begin{cases} \theta_2 - \theta_1 = \alpha_1 - (\alpha_2 + \alpha_3) = \theta_0 \\ \frac{sin\theta_1}{l} = sin\alpha_1 r = k_1 \\ \frac{sin\theta_2}{l} = sin\alpha_2 r = k_2 \end{cases} \tag{4}$$

$$\Rightarrow \theta_1 = arctan\frac{sin\theta_0}{\frac{sin\alpha_2}{sin\alpha_1} - cos\theta_0}, \theta_0 = \alpha_1 - (\alpha_2 + \alpha_3)$$

或 $\theta_1 = 90°$。

若 B 的极坐标为 (100,β)，则该种情况下，未知点的定位坐标为 (100×$\frac{sin\theta_1}{sin\alpha_1}$,$\beta$−($\pi$-$\theta_1$-$\alpha_1$))。

由于 θ_1 有四种不同的表达方式，故未知点仍有四个可能的坐标位置。

5.1.2.1 最终位置的确定

由于我们无从知道 D 点究竟落在哪个区域，故四个点的坐标均有可能。然而题目中提及被动接收信号无人机所在位置仅略有偏差，故我们考虑将求解出来的四个坐标分别与理想坐标求欧几里得距离，将其中距离最短的点作为该无人机所在的位置。

距离公式计算方式如下：

设 D 点的理想坐标为 (100,γ_0)，求解出来的其中一个坐标为 (100×$\frac{sin\theta_1}{sin\alpha_1}$,$\beta\pm(\pi$-$\theta_1$-$\alpha_1$))，则：

$$s = \sqrt{100^2 + (100 \times \frac{sin\theta_1}{sin\alpha_1})^2 - 2 \times 100 \times (100 \times \frac{sin\theta_1}{sin\alpha_1}) \times cos(\gamma_0 - \beta \pm (\pi - \theta_1 - \alpha_1))}$$

5.1.3 几何模型的相关检验

为了验证该位置确定方式的严谨性，我们进行了相关数据的模拟。随机选取两个编号的无人机作为发射源无人机，另外七个编号的无人机作随机扰动，使其略微偏离理想位置。由于我们采用的为极坐标系，所以给出一个扇形区域的误差限，但为了跟实际情况更加匹配，通过调整极径和幅角，使扇形区域尽可能接近方形区域。

经过相关计算，我们给出极径误差为 $\pm 5m$，幅角误差为 $\pm 3°$，从而使得随机误差在 10% 以内，且浮动区域约为一个方形，如图7所示。

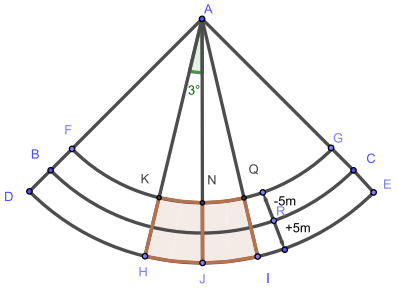


图 7: 误差范围示意图

模拟方式为在 9 个编号点中选取其中 3 个，两个点为发射点，一个点为接收点，给接收点设置一个随机扰动，得到一组模拟真实数据。计算接收点与发射源间的方向信息后，将其作为输入，用上文所建的模型进行求解后，得到一组求解数据。用模拟真实数据与求解数据进行比较，以此判断模型的准确性。

选取方案总共有 $\binom{3}{9} \times 3 = 252$ 种，将其全模拟并且在随机扰动 10^5 次后，发现模拟真实数据与求解数据间的误差在小数点 10 位之后，考虑为机器误差。由于数据量过大，故不将具体数据放入正文及附录中，具体内容可见于支撑材料。

5.1.4 发射源的选取

共有四类发射源的选取方式，分别与原点的夹角为 $\pm 40°$，$\pm 80°$，$\pm 120°$，$\pm 160°$。

不断扩大误差范围，直至扩大到误差为 30%（即极径误差限扩大为 $\pm 15m$，幅角误差限扩大为 $\pm 3°$ 时，出现“报错现象”。“报错现象”的定义为模拟真实位置与求解位置之间的距离超过机器误差 eps（根据经验，约为 10^{-8}）。

在上部分的模拟中，这四类情况我们均有涉及，现为扩大数据量，增加实验可信度，每轮模拟次数调整为 10^7 次。其中，发射源的圆心角为 $40°$ 时，报错概率较小，约为其他几类情况报错概率的 $\frac{1}{2}$。

运行三轮的数据基本保持稳定，具体结果如下图8所示：

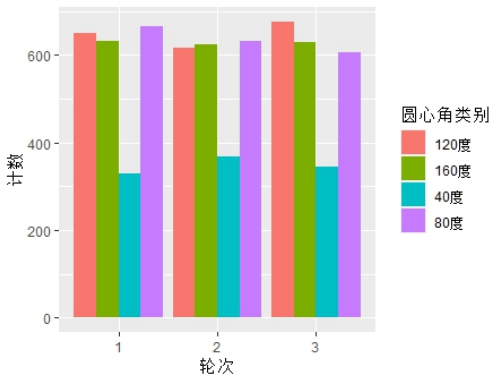


图 8: 不同发射源圆心角的报错频次图

故在进行发射源的选取时，可以尽可能多考虑圆心角为 40° 的两个无人机作为发射源。

5.2 问题二模型的建立与求解

在信号源编号未知的情况下，每次多引入一个信号源，只能额外得到以接受信号无人机为顶点的两个有效角度信息。从几何求解角度，其不足以得到定位所需的极径、幅角。于是我们考虑利用已知角度信息先确定至少一架发射信号无人机的编号，即可把该问题等同于问题一，利用相同方法定位。

5.2.1 角度区间定位模型

5.2.1.1 角度范围划定

考虑以待定位无人机理想位置为顶点、FY01 与另一未知编号无人机所发射信号为两边形成的圆周角，圆周上相邻的发射信号无人机对应的该角度有 20° 的变化量。鉴于待定位无人机位置只是略有偏差，故不同发射信号无人机对应的该角度也只是相对理想圆周角略有改变，不同的信号源无人机形成的圆周角仍可认为处于不同的跨度为 20° 的角度区间中。

对待定位无人机，根据每 20° 对应一个信号来源无人机，对接受到的信号的角度进行线性区间划分。再根据实际测得的角度信息落入的角度区间，找到其对应的发射信号无人机，即可推定信号源无人机的编号。

又由于等长的弦对应的圆周角相等，故由上会得到两个关于过 FY01 的径向对称分布的均满足该角度情况的信号源无人机编号。

例如在 FY04 作为待定位无人机、FY03 作为一发射信号无人机的情况下，接受到的信号近似圆周角为 38°，此时对各个不同信号源无人机位置，有对应角度划分区间如下表；

表 2：各编号无人机作信号源时对应角度区间

无人机编号	角度区间 (°)	无人机编号	角度区间 (°)	无人机编号	角度区间 (°)
FY02	[0,30)	FY03	[30,50)	FY05	[70,90]
FY06	[70,90]	FY07	[50,70)	FY08	[30,50)
FY09	[0,30)				

该表由 FY01 作为起始零度点，向两侧每隔 20° 进行划分得到。其中因为 FY01 作为已知的固定发射源存在，故将较小的角度区间直接划分给两侧的 FY02、FY09。同时我们将可能的测得角为钝角情况考虑为一对优弧、劣弧对应的一对对称无人机情况，故表中直接考虑其补角即可。

在该情境下，测得的 38° 落在了 $[30, 50)$ 划分区间，故由表可知其最可能的信号发射来源为 FY03、FY08 这一对无人机。

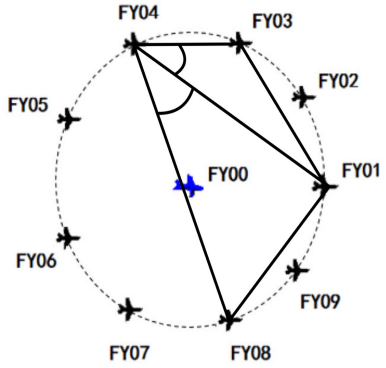


图 9: 接收信号角度示意图

5.2.1.2 对称点的判断

当将获得的方向信息对应在相应的角度区间上时，我们可以判断未知发射源无人机的可能编号。编号的可能性一般有两种且关于 FY01 号无人机的位置对称，为了判别发射源无人机的真实编号，我们对其进行机理分析。

设需要判别的两个编号为 c_1，c_2，其中 $c_1 + c_2 = 11$，在圆周上均匀分布着 9 个点，1 个点为已知发射源 FY01，2 个点为可能发射源。这三个点上无人机的位置不存在偏差，根据对称特性，2 个可能发射源与 FY01 中相隔点的数量相同，将这些相隔点定义为区内点。除去这三类点后，圆周上一定还存在偶数个点，并将这些点定义为区外点，具体定义如下图10。接收信号的无人机的真实位置与这些点略有偏差。

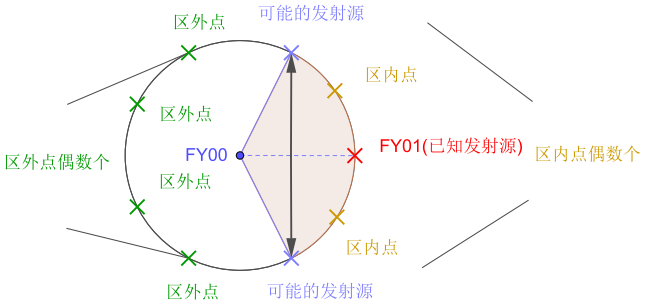


图 10: 点的定义图

接收点存在的位置有两种可能，一种为区内点附近，一种为区外点附近。

为了更清晰地进行阐释，下面给出了两个典型情况的举例（左图为接收点在区内点附近，右图为接收点在区外点附近）。设接收点为 FY02/FY06 附近略有偏差的点，可能发射源为 FY03 和 FY08。

令 FY00 所在点为 O，FY01 为 P，可能发射源分别为 F_1、F_2，接收点为 J，如下图1112所示。

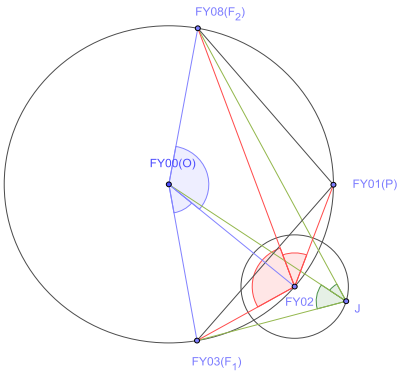


图 11: 区内点情况

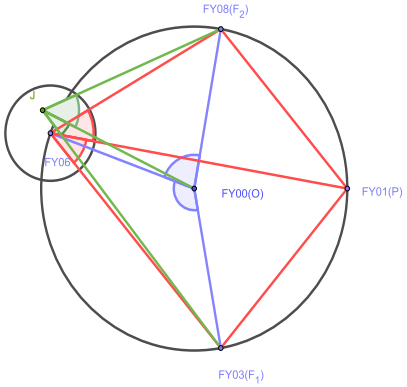


图 12: 区外点情况

由左图可以看出，P 点等分了发射源包含的区间，由于接收点的位置仅略有偏差，我们不妨假设它就在 J 点上。显然，接收点仅能落在等分区间的其中一个上，区间内与两个可能发射源圆心角相同的点仅有 P 点，因此接收点与不同发射源之间的圆心角一定不同。

由右图可以看出，F_1，F_2 之间不包含 P 点的弧被区外点划分成了奇数段（包括 F_1，F_2 共有偶数个点，首尾不相连，组成奇数个区间），J 点将这奇数个区间再次进行划分，分为奇数段的区间及偶数段的区间。显然这两段区间对应的圆心角明显不同。

故我们采用 FY00 和可能发射源无人机给出的方向信息对编号进行鉴别（即 $\angle OJF_1$ 或 $\angle OJF_2$），由于 ΔOJF_1 及 ΔOJF_2 均近似为等腰三角形，故在圆心角不同的前提下，方向信息也不相同。比较真实得到的方向信息与两个理论角度，将与真实值更近的理论角度判为真，选择其对应的发射源为真实编号数。

5.2.2 结果展示与误差分析

算法流程大致分为以下两步：

Step 1: 根据前文所建模型得出未知发射源的编号

Step 2: 利用第一问的算法确定接收信号无人机的所在位置

与第一问相同，控制接收端的偏移误差在 10% 以内，构造一个近似为方形的扇形区域，即极径误差为 $\pm 5m$，幅角误差为 $\pm 3°$。

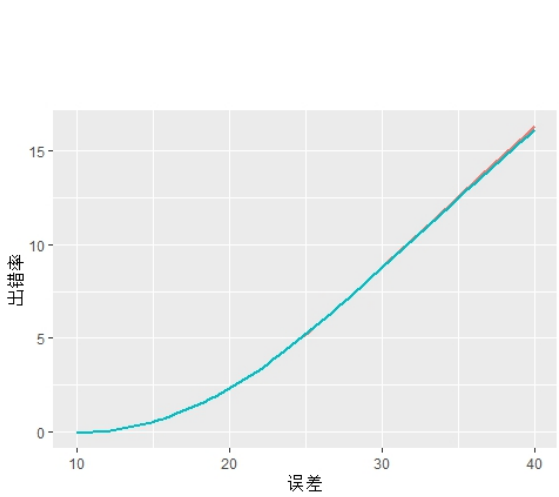
模拟方式为在 9 个编号点中选取其中 2 个，一个点为发射点，一个点为接收点，给接收点设置一个随机扰动，得到一组模拟真实数据。计算接收点与发射源间的方向信息后，将其作为输入，用算法进行求解后，得到算法给出的发射源求解编号以及接收端求解位置。将其与模拟真实数据进行比较，以此判断模型的准确性。

选取方案总共有 $\binom{2}{9} \times 2 = 72$ 种，将其全模拟并且在随机扰动 10^6 次后，发现报错率（即编号选取出错概率）为 0。

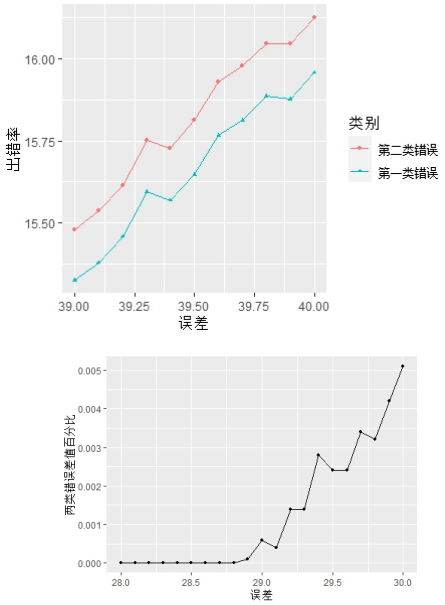
为了检验模型可承受的误差范围，我们以 0.1% 的间隔不断扩大偏移范围，发现在误差超过 12% 之后，开始有“报错现象”的出现。

“报错现象”出现的原因有两类：第一类为未知编号判断错误，第二类为定位错误（可能源于未知编号判断错误，也可能是第一问的算法定位错误），也就是说，第一类错误的发生必将导致第二类错误的发生。

两类错误的具体报错概率如下图17所示:



(a) 两类错误整体趋势



(b) 误差为 39%-40% 两类错误的差异及差异分界点

图 13: 两类错误报错概率折线图

由图中所给信息可以看出，当接收点偏移的误差大于 12% 时，接收点定位出现错误；而在误差为 12%-29% 时，定位错误完全由编号选取失误造成；当误差接近 29% 时，由第一问算法求解造成的误差开始逐渐增大，但当误差范围达到 40% 时，依然远远小于编号选取错误造成的定位误差。

通过对比分析误差来源的类型，该问所建的模型对“略微偏移”的要求较高。一旦需要定位的无人机偏移位移较大，定位模型有一定概率会失效。

5.2.2.1 误差来源

该模型编号选取错误主要来自两个方面：第一是角度划分区间确定可能发射源选取错误，第二是对称点选取错误。

当接收端无人机离理想位置偏离较大时（超过 $\pm 6m$ ），可能会不符合我们人为设定的角度划分区间及对称点的角度范围。

为了使无人机定位有更高的精度，我们对模型进行进一步的修正。

5.2.3 投票机制模型的建立

前文模型建立过程中，一个较大的误差来源为角度划分区间过于死板，虽这样划分能在得到方向信息后直接确定未知发射源的编号（对称两点），较为简易。但当接收点偏移位移较大时，如图14所示。

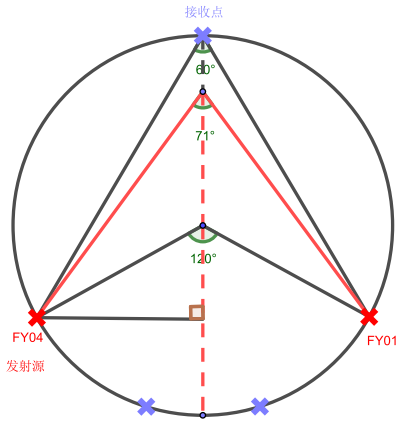


图 14: 错误判断编号示例

在这种情况下，本应是真正发射源的 FY04，被误判成了 FY05/FY06，导致错误。

因此，我们决定在接收点的理想位置附近划定误差圆，通过误差圆来确定角度范围，但由于误差圆的半径选择依然较为主观，故我们选择用一系列大小不一的误差圆进行投票，以此确定未知发射源的编号。

5.2.3.1 角度范围的求解

由题意，无人机的真实位置在自身编号的理想位置附近，故我们以理想位置为中心作出误差圆。每一个可能的发射源（除无人机自身编号与 FY01 外，共有 7 个）均能根据误差圆得到一对判定的角度范围，具体求解如下：

考虑待定位无人机 J 在一以理想位置 Q 为圆心、半径为 r 的误差圆内，其与 FY01(P)、未知编号无人机 (F) 间形成的 $\angle PJF$ 的，以及与 FY01(O) 形成的 $\angle PJO$ 的大小为随 J 位置变化的函数。在确定二者关于 J 位置变化的区间后，即可对应找到同时满足两个角度区间的信号源无人机。

在求解两函数的变化区间时，可将问题考虑为一圆心在给定弦中垂线上变动，求变动的圆中该弦对应的圆周角的大小范围的等价命题。因为对给定弦，其所在的圆半径越大，对应的圆周角越小；因此当该动圆变动到恰与误差圆外切、内切时，半径取到最小、最大值，角度大小也相应取到最值。如在求解 $\angle PJF$ 大小范围时，取最值的具体情形见下图：

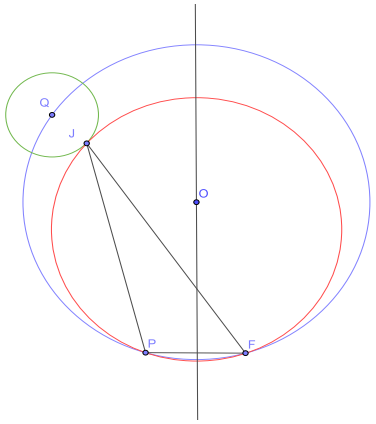


图 15: $\angle PJF$ 最大值示意图

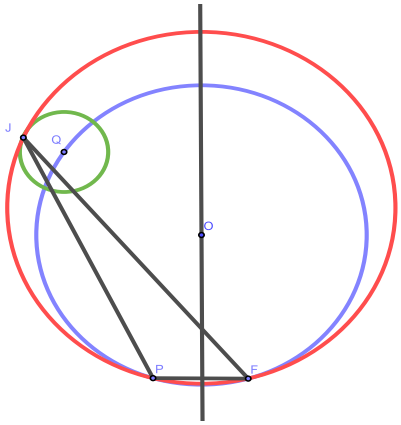


图 16: $\angle PJF$ 最小值示意图

由于圆的凸性，两个角度大小的函数的最值一定在圆周上取到。因此虽然上述理论方法给出了准确求法，但在实际计算过程中，为了方便实现，我们采用了蒙特卡洛方法对其最值进行了可靠的估计。计算时先随机生成一个角度值，得到误差圆上对应角度的点的坐标，代入求解其对应的 $\angle PJF$ 的大小。通过数次数值实验，我们采用 10^3 次随机采样的最值来近似该函数的最值，经检验其具有良好的精确度。

5.2.3.2 误差圆系的构造

由于接收端的真实位置是未知的，我们无法得出无人机具体被框定在哪个误差圆内，因此我们构造了半径为 0.1m-40m，间隔为 0.1m 的 400 个同心圆，圆心均为接收信号无人机的理想位置。

对每个误差圆均可求解得到可能发射源给出的一对角度范围，将其与接收端得到的方向信息进行比较，以此为基础进行投票。

5.2.3.3 投票机制

在正式投票前，我们可以得到一张已知信息表，其中包含误差圆 Q 的半径（范围为 0.1m-40m）；误差圆的圆心编号 I（范围为 2-9）；可能发射源的编号 F（范围为 2-9）；两个角度范围：包含 FY00(O) 点的角度范围 (O_{min}, O_{max}) 及包含 FY01(P) 点的角度范围 (P_{min}, P_{max})。

由于数据条数过大（约有 $400*8*7*4=89600$ 个实数），该表暂不放在论文或支撑材料中，可使用相关代码输出该数据。

我们让每个接收点理想位置 I 为圆心的误差圆根据接收点得到的角度信息进行投票，根据票数多少选择最可能的发射源 F_1。

设真实的发射源为 $Ftrue_1$，则当接收点 J 得到了 $\angle OJFtrue_1$ 和 $\angle PJFtrue_1$ 的角度信息后，对于每个发射源 F，它的得票数 $vote_F$ 可由下列等式计算得到：

$$vote_F = \sum_Q [V_{Q,I,F,Omin} \le \angle OJFtrue_1 \le V_{Q,I,F,Omax}][V_{Q,I,F,Pmin} \le \angle PJFtrue_1 \le V_{Q,I,F,Pmax}] \quad (5)$$

即 400 个误差圆中，每个误差圆都根据接收点得到的角度信息是否在每个发射源对应的角度区间内，可以给最少 0 个（角度信息不在任何发射点对应的角度区间内），最多 7 个（角度信息在除了已知发射源 P 和接收点理想位置 I 的所有发射源对应的角度区间内）发射源投票，最后我们将票数最高的发射源作为 F_1，即:

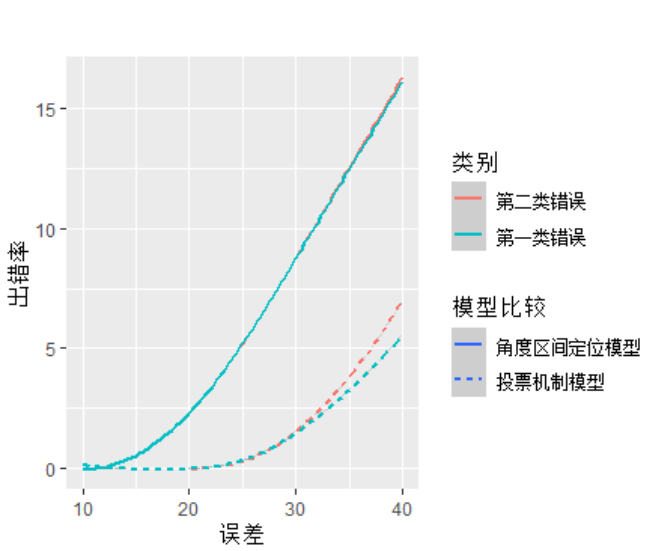
$$F_1 = \arg\min_F vote_F \quad (6)$$

5.2.4 结果展示

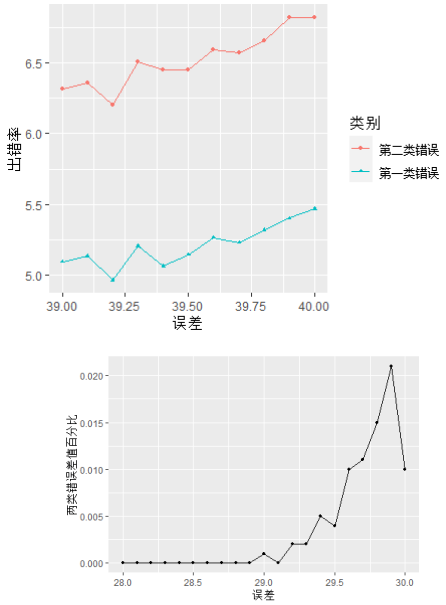
为了检验模型的正确性，我们同样进行模拟检验。与上文中角度区间定位模型的模拟方式相同。即：在 9 个编号点中选取其中 2 个，一个点为发射点，一个点为接收点，给接收点设置一个随机扰动，得到一组模拟真实数据。计算接收点与发射源间的方向信息后，将其作为输入，用算法进行求解后，得到算法给出的发射源求解编号以及接收端求解位置。将其与模拟真实数据进行比较，以此判断模型的准确性。

同样以 0.1% 的间隔不断扩大偏移范围，发现在误差超过 20.7%（极径误差为 $\pm 10m$，幅角误差为 $\pm 6°$）之后，才开始有“报错现象”的出现，相较前一个模型有明显提升。

对上文中提到的两类错误重新作图17分析如下:



(a) 两类错误整体趋势



(b) 误差为 39%-40% 两类错误的差异及差异分界点

图 17: 两类错误报错概率折线图

由图中所给信息可以看出，当误差大于 21% 时，接收点定位出现错误；而在误差为 21%-29% 时，定位错误完全由编号选取失误造成；但整体而言，用投票机制构建的模型求解的效果远远好于角度区间定位模型。

现在我们只增加了一架未知编号的发射源无人机提供方向信息，当误差达到 40% 时，依然有近 7% 的定位错误的可能。为了进一步提高精度，我们可以考虑多引入一架无人机。

5.2.4.1 进一步的思考

虽然投票机制令计算接收点位置错误的概率大大降低，但在接收点随机误差在 40% 的范围内浮动时，仍有 6.821% 的定位错误，而其中大部分（5.469%）是由于发射点编号确定出错导致的，故需要进一步优化模型，提高计算发射点编号的正确率。

考虑加入第二架位置编号的发射信号的无人机 $Ftrue_2$，接收点 J 还将得到 $\angle OJFtrue_2$ 和 $\angle PJFtrue_2$ 的角度信息,那么我们可以给所有可能的发射源对投票,以确定最可能的发射源对 F_1, F_2。

那么利用类似之前的投票方法，我们可以用以下等式计算每个点对的得票数：

$$vote_{F,F'} = \sum_{Q}[V_{Q,I,F,Omin} \le \angle OJFtrue_1 \le V_{Q,I,F,Omax}][V_{Q,I,F,Pmin} \le \angle PJFtrue_1 \le V_{Q,I,F,Pmax}]$$
$$[V_{Q,I,F',Omin} \le \angle OJFtrue_2 \le V_{Q,I,F',Omax}][V_{Q,I,F',Pmin} \le \angle PJFtrue_2 \le V_{Q,I,F',Pmax}] \tag{7}$$

即 400 个误差圆中，每个误差圆都根据接收点得到的角度信息是否在每个发射源对应的角度区间内，可以给最少 0 对，最多 $7*6=42$ 对发射源投票，最后我们将票数最高的发射源作为 F_1, F_2，即:

$$F_1, F_2 = \arg\min_{F,F'} vote_{F,F'} \tag{8}$$

为了减少通信资源浪费，我们将尽量使用一个未知编号发射源进行定位，当定位失败后，才考虑使用第二个未知编号发射源；但由于实际应用时无法得知真实坐标，故我们不能通过计算真实位置 J 和算法结果 J_0 之间的差值 $|J-J_0|$ 来判断是否定位成功，故我们通过算法结果导出应得的角度信息 $\angle OJ_0F_1$ 和 $\angle PJ_0F_1$ 和真实数据比较来判断是否定位失败，即当 $|\angle OJ_0F_1-\angle OJFtrue_1|>eps$ 或 $|\angle PJ_0F_1-\angle PJFtrue_1|>eps$ 时判断为定位失败，再调用二轮投票模型来进行进一步的定位。

为了验证模型正确性，我们用和上文角度区间定位模型与投票机制模型的模拟方法相同的模拟方式进行模拟。将三种模型的报错情况作图18如下：

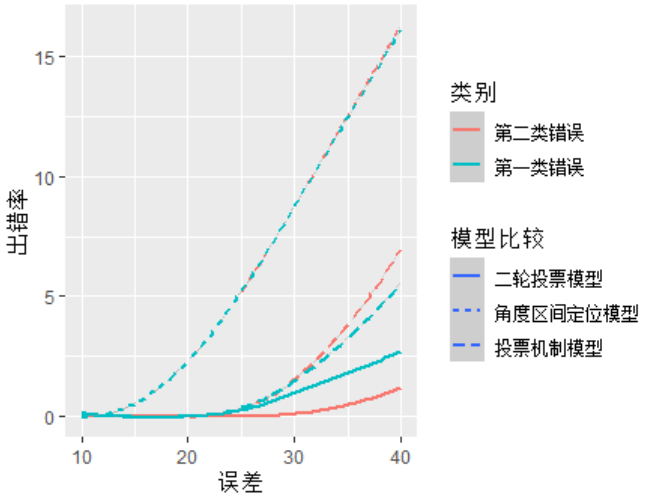


图 18: 三种模型报错概率折线图

不难看出二轮投票模型的报错率明显低于其他两个模型，甚至能够在计算出的发射点不完全正确的情况下得出正确的定位信息（数据表现为第一类错误率大于第二类错误率），模型优化效果明显。

在模拟时，我们也记录了两种判断定位失败的方法判断的结果，发现在所有模拟中两种方法计算得出的定位失败率都相同，故可以证明用角度判断定位准确度的正确性。（具体结果可以查看支撑文件）

5.3 问题三模型的建立与求解

由于该问的整体流程和思路较为复杂，故绘制流程图19如下：

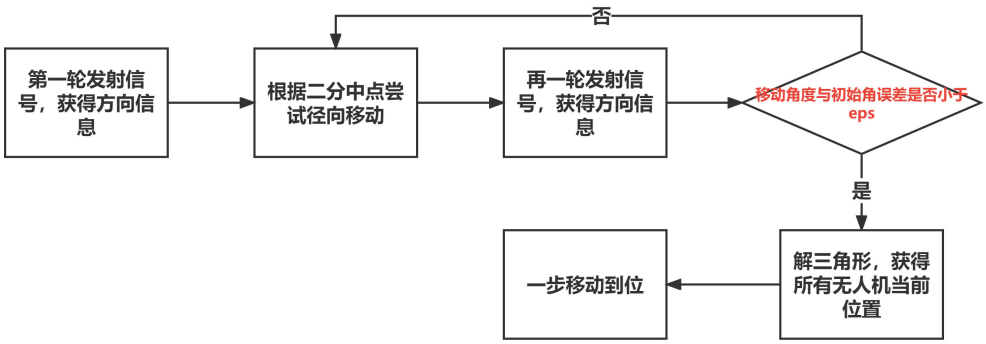


图 19: 问题三流程图

5.3.1 坐标系的建立

由于现实场景中无人机水平方向的操纵命令往往是经由进退、左右式给出的，因此在本题中考虑建立坐标系描述无人机的运动过程时，选取了更贴近其运动形式的直角坐标系。

同时为了便于描述无人机的运动，我们进一步选取无人机的“前进”方向作为 x 轴的正方向，其“左侧”方向作为 y 轴的正方向，并将原点定在位于机群中心的 FY00 处。同时为了便于后续整体队形的调整，我们将 FY01 的理想位置定于 (100,0)。而由题目所给图片机群的机头朝向，我们假定机头朝向是机群前进方向，设其为 x 轴方向。再根据题目给出的初始时刻无人机位置仅略有偏差，在此坐标系下 FY01 的实际位置与设定的理想位置 (100,0) 相距不远，故所建立的坐标体系合理。

5.3.2 信号发射机次的选取

为了方便每驾无人机的幅角方向的确定，我们考虑先通过无人机之间的信号传输得到每架无人机 J 与 OP 的夹角 $\angle POJ$。具体地我们选取如下的信号发射机次方案：

表 3：各轮次选取的进行信号发射的无人机编号

	轮次一	轮次二	轮次三	轮次四
发射信号无人机编号	(0,1)	(0,2,3,4)	(0,5,6,7)	(0,8,9)

在第一轮次中，选取 FY00、FY01 作为发射信号的无人机，其余无人机均作为接受信号的无人机。由此可以得到所有以其余无人机位置为顶点、与 (FY00,FY01) 两点构成的 $\angle OJP$ 的角度值。

后续三轮次中依旧保留了 FY00，再选取了除 FY00、FY01 外其他的无人机依次来发射信号。其中重点考虑 FY01 接收到的角度信息，即以其为顶点的 $\angle OPJ$。

结合各轮次信息，对除 FY00、FY01 外的无人机 J，在 ΔJOP 中即可求得圆心角 $\angle JOP$。

5.3.3 二分移动路径法

首先我们需要确定圆周上的每架无人机到原点的连线和 x 轴的夹角大小，现在以编号为 $FY02$ 的无人机为例说明如何确定该夹角大小。

设 $FY1, FY2$ 分别为 $FY01$ 和 $FY02$ 的理想位置，$FY1^{'}, FY2^{'}$ 分别为 $FY01$ 和 $FY02$ 的真实位置，坐标原点为 O, 则 α 为 $OFY1^{'}$ 和 $OFY2^{'}$ 两边的初始夹角。设 $FY2^{''}$ 是 $FY02$ 调整后的位置，则 $\alpha^{'}$ 为 $OFY1^{'}$ 和 $OFY2^{''}$ 的夹角。若有 $\alpha = \alpha^{'}$，说明 $FY2^{''}$ 在 $OFY2^{'}$ 的延长线上。令 $FY2^{''}$ 在以 $FY2^{'}$ 为圆心，常数 r(具体实现时设置为 $1m$) 为半径的圆上调整。尽管我们初始时无法知道 $FY2^{'}$ 的精确位置，但通过 r 和 θ 容易计算 $FY2^{'}$ 到 $FY2^{'}$ 的变化量，故可以使用形式为 $(\Delta x, \Delta y)$ 的调整指令使无人机在圆周上调整。

设 $FY2^{''}FY2^{'}$ 与 x 轴的夹角为 θ，则 θ 和 $\alpha^{'}$ 之间存在一个函数映射关系 $\alpha^{'} = f(\theta)$，那么 $FY2^{'}$ 的方向确定问题就转化为了 $\alpha = f(\theta)$ 的方程求解问题。根据 $FY02$ 的理想位置划定 θ 的大致区间 $[40^{\circ} - 20^{\circ}, 40^{\circ} + 20^{\circ}]$ 后，不难发现在此范围内 $f(\theta)$ 是单调的，那么我们就可以使用二分法求解 $\alpha = f(\theta)$。

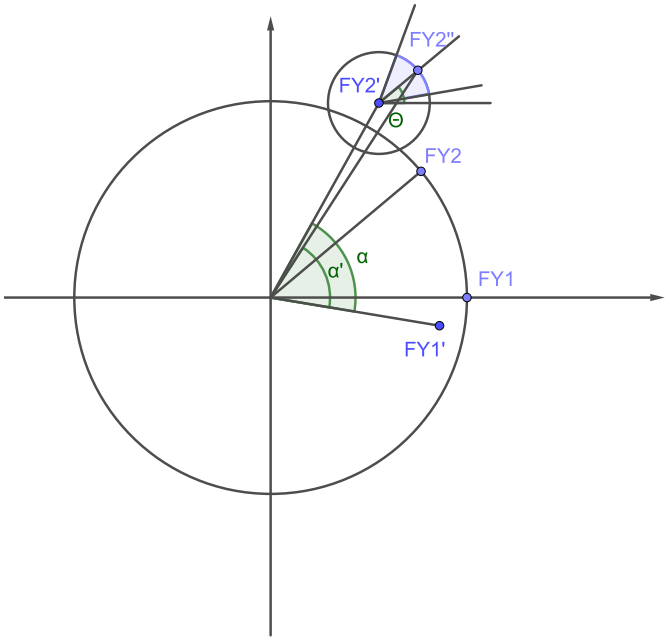


图 20: 二分移动路径法示意图

我们可以对其他编号的无人机根据其理想位置确定不同的 θ 范围，在各自的范围内分别二分移动路径，最终确定各自位置到原点的连线和 x 轴的夹角大小。

5.3.4 结果求解

当二分移动结束后，可以通过前几次移动的方案求出该编号无人机当前所在的位置。位置坐标在极坐标系下表示，极坐标系的极轴与原平面直角坐标系的 x 轴重合。

a. 幅角的计算

设无人机共移动 N 次到达期望位置，第 i 次无人机的移动指令为 $(\Delta X_i, \Delta Y_i)$，最终状态下无人机在水平方向上移动了 $\sum_{i=1}^{N} \Delta X_i$ 的距离，在竖直方向上移动了 $\sum_{i=1}^{N} \Delta Y_i$ 的距离。

所以，这 N 次移动等价于从初始点到最终点进行一次移动 $(\sum_{i=1}^{N} \Delta X_i, \sum_{i=1}^{N} \Delta Y_i)$。

由于二分法的目的是使最终状态无人机的位置幅角等于初始状态无人机的位置幅角。故幅角计算公式为：

$$\theta = arctan(\frac{\sum_{i=1}^{N} \Delta Y_i}{\sum_{i=1}^{N} \Delta X_i}) \tag{9}$$

b. 极径的计算

选取相邻两个无人机求解三角形，如图21所示：

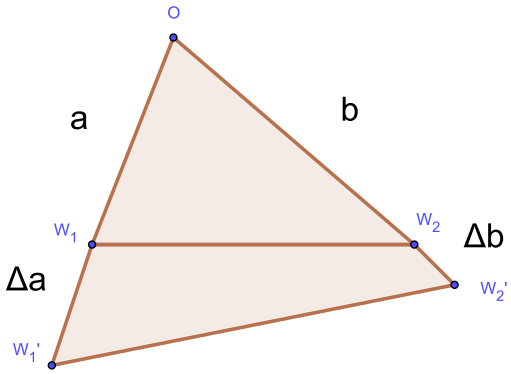


图 21: 三角形求解示意图

ΔOW_1W_2 为两无人机在初始位置时形成的三角形，$\Delta OW_1'W_2'$ 为两无人机在最终位置时形成的三角形。经过第一轮的信号发射，可以得到方向信息 α_1，β_1; 经过最终一轮的信号发射，可以得到方向信息 α_2，β_2。

其中 a 为 W_1 无人机在初始时的极径，b 为 W_2 无人机在初始时的极径。Δa 为 W_1 无人机最终位置与初始位置的距离，Δb 为 W_2 无人机最终位置与初始位置的距离。

利用正弦定理解三角形可得：

$$\begin{cases} b = \frac{\Delta b - \Delta a}{\frac{sin\alpha_1}{sin\beta_1} - \frac{sin\alpha_2}{sin\beta_2}} \\ a = \frac{\Delta b - \Delta a}{1 - \frac{sin\alpha_2 \cdot sin\beta_1}{sin\alpha_1 \cdot sin\beta_2}} \end{cases} \tag{10}$$

其中，

$$\Delta a = \sqrt{(\sum_{i=1}^{N} \Delta X_{i,1})^2 + (\sum_{i=1}^{N} \Delta Y_{i,1})^2}$$

$$\Delta b = \sqrt{(\sum_{i=1}^{N} \Delta X_{i,2})^2 + (\sum_{i=1}^{N} \Delta Y_{i,2})^2}$$

则 W_1 无人机现在所在的位置 W_1' 的极径为 a+Δa，W_2 无人机现在所在的位置 W_2' 的极径为 b+Δb。

获得所有无人机在极坐标系上的坐标后，即可将其按照理想位置一步移动到位。

5.3.5 结果展示

为使二分法尽可能精确（控制幅角误差在 10^{-6} 左右），我们设定整个流程循环 20 次，即共经历 21 轮次信号发射，其中每轮次包含 4 次不同发射源的信号发射。故至多需要 84 次不同发射源的信号发射，即可实现有效定位。

题目要求无人机在最终需调整到理想位置，使得 9 架无人机最终均匀分布在圆周上。本文将理想位置设定为如表 1 所示的理想位置坐标。

5.3.5.1 方案展示

由于方案中信号发射的次数较多，故完整方案无法在正文中展示，而放于附录中。方案包括每一轮发射源的选择以及各编号无人机的移动距离。

计算发现，累积误差（9 个点求解位置与理想位置的距离和）为 $1.60328 * 10^{-4}m$，说明求解效果良好。

5.3.5.2 模型的进一步检验

由于题中所给的初始点的位置相较理想位置的误差较小，故为了对模型进行更好的检验，我们考虑扩大误差观察求解效果。

将误差限扩大到 30%（即极径误差为 $\pm 15m$，幅角误差为 $\pm 9°$）进行模拟。

模拟方法为：在 9 个理想位置附近各框定 30% 的误差区域，每一个误差区域中随机选取一个点作为该区域编号无人机的初始位置。将其作为输入代入求解具体调整方案。

模拟 10^6 次的试验后，我们发现累积误差均控制在 10^{-3}m 以内，说明在误差较大的情况下也能进行较优的求解，使 9 个偏移位置点移动到理想位置。

5.4 问题四的分析解决

我们尝试将求解问题三的方法推广运用到问题四中。考虑在锥形编队情景下，仍选取一对无人机位置固定不动、形成一条固定的信号边；对其他无人机尝试移动时，寻找控制其与该边夹角不变的移动方向，即可得到其初始位置与坐标轴的方向角。具体实现过程中，仍可沿用三题中所述的二分查找方法，通过在一小段轨迹圆弧中不断尝试来得到所求方向。

5.4.1 坐标轴的建立与测量基准点无人机的选取

建立坐标轴的整体考量与问题三类似：选取 FY01 作为坐标原点；假定图中机群机头朝向为其前进方向，定为 x 轴正方向，再以右手螺旋形式建立 y 轴即可，如图22所示。验证知在该坐标系中各无人机的位置均能较简洁地表示出来，坐标系建立合理。

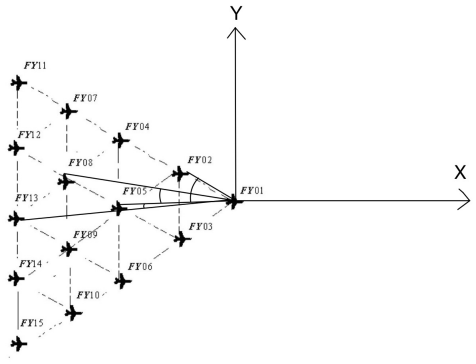


图 22: 坐标系建立示意图

首先我们考虑固定 FY01 作为测量基准点。因为 FY01 处于编队的顶角位置，其他无人机均在其周围一个较小的角度范围内，能使测量得到的角度在一个大小合理的角度区间。接着我们考虑固定 FY05 作为另一基准点。因为大部分无人机均对称分布在 FY01-FY05 轴线两侧，能保证后续二分时大部分点所取的轨迹区间使得其与固定边的夹角单调变化，即使问题拥有可二分性。

同时初始轮次二分进行各个无人机方位角的测定时，FY13 虽不是基准点、但也先固定不动。这是考虑到 FY13 位于上述选定的 FY01-FY05 轴线附近，所选取的查找轨迹区间可能穿越轴线，使其丧失了用于二分的角度单调性，不能得到其方向角的真实值。我们将在后续轮次中单独考虑为 FY13 定位的问题。

5.4.2 二分移动路径法

在锥形编队队形下，我们可以通过控制其他无人机到 $FY01$ 的连线与 $FY01-FY05$ 这条边的夹角大小，二分确定其他无人机的方向，具体二分方法和问题一的第 3 小问的方法类似，但由于 $FY01$ 可以作为接收点，我们可以直接得到夹角的角度信息，而不是通过三角形内其他两个角的角度，间接计算夹角。

5.4.3 结果求解

由上文阐述的发射信号与移动方案可知，在二分循环中 FY01、FY05、FY13 均未移动。将 15 架无人机的当前位置分成三类计算：

（1）FY01 编号无人机为测量基准点，假设其初始位置即为理想位置，则其当前位置为原点 (0,0)；

（2）除 FY01、FY05、FY13 这三个点外，其余点按第三问中的方式进行求解。其中:

a. 幅角计算公式

$$\theta = arctan(\frac{\sum_{i=1}^{N}\Delta Y_i}{\sum_{i=1}^{N}\Delta X_i})$$

b. 极径的计算

选取 FY01，FY05 以及所需求解的点的位置解三角形，即在图21的情况下，令 Δb=0。

故极径的计算公式为：

$$a = \frac{\Delta a}{\frac{sin\alpha_2 \cdot sin\beta_1}{sin\alpha_1 \cdot sin\beta_2} - 1} \tag{11}$$

则 W_1 无人机现在所在的位置 $W_1^{'}$ 的极径为 a+Δa。

（3）FY13 编号的无人机由于无法用二分法求解的限制，需要借助其他手段进行位置求解。

再发射一次信号，分别为 FY01(O)，FY13(Q) 以及在 X 轴上方的无人机（例如 FY02(P)），得到相应的方向信息 $\angle POQ$。

a. 幅角的计算

由于 P 点与 X 轴的夹角已知，P 点与 Q 点的夹角已知，且 Q 点在 P 点的逆时针方向，故幅角计算公式为：

$$\theta_Q = \theta_P - \angle POQ \tag{12}$$

b. 极径的计算

由于 P 点与原点间的距离已知，故应用正弦定理可知 Q 点的极径：

$$OQ = OP \times \frac{sin\angle OPQ}{sin\angle OQP}$$

获得所有无人机在极坐标系上的坐标后，即可将其按照理想位置一步移动到位。

5.4.4 结论说明

当需要定位的无人机偏差过大时，如 $FY08$ 偏差到了 $FY01-FY05$ 连线下方，则会使其丧失可二分性，可能导致求解失败。当偏差较小时，则可以在共 20 轮询问和调整后，将每个无人机的角度偏差降低到 10^{-5} 以内。

6 模型的评价与推广

6.1 模型的优点

- 定位精度高，问题一的定位误差可基本控制在 $10^{-8}m$ 内，问题二三的定位误差和累积误差可基本控制在 $10^{-4}m$ 内。
- 使用无人机接收到的角度信息验证定位准确性，可以在实际应用时进行检验，不需要得知真实坐标，适用性强，且在计算机模拟时检验结果和用真实坐标的检验结果相同，正确性高。
- 投票机制让定位错误率大大下降，同时尽量使用了少的发射无人机进行定位，且投票机制衍生出的二轮投票机制可以进一步延伸，得到使用三架甚至更多未知编号的发射无人机进行定位的多轮投票模型，进一步提高定位精度和正确率。

6.2 模型的缺点

- 当需要定位的无人机距离其理想位置偏差较大时，模型可能出现定位错误的情况，此时可以考虑加入更多发射源。
- 基准无人机需要处理的角度信息远多于其他无人机，对其计算性能和精度要求较高。
- 在投票模型中计算误差圆对应的角度范围时使用了较为方便实现的蒙特卡洛方法，精度较低且复杂度较高，可以考虑改用论文提到的计算几何相关的计算方法。
- 在最后一次调整到接近理想位置之前，在前期二分调整时，各无人机可能会先远离理想位置。

6.3 模型的推广

问题一中的定位模型可以推广到用平面上任意 3 个已知位置的发射源及接收点得到的角度信息来定位一个已知大致位置的接收点，不必局限在圆周附近进行定位。

问题三和问题四中先二分确定角度，再解三角形确定距离以定位每个无人机和基准无人机的相对位置的思路也可以拓展到其他编队队形的调整上。

参考文献

[1] 赵骁飞. 单站纯方位无源探测定位的若干技术的研究 [D]. 上海交通大学,2007.

附录

附录说明:

1,**readme**：支撑文件说明

2,**geometry.h**：其他程序需要的几何相关的头文件

3,**task1.cpp**：对问题一的模型进行多次模拟验证，输出报错统计信息

4,**task2_simple.cpp**：对问题二的角度区间定位模型进行多次模拟验证，输出需要定位的点在不同误差范围下的报错统计信息

5,**task2_vote.cpp**：对问题二的投票机制模型进行多次模拟验证，输出报错统计信息

6,**task2_vote2.cpp**：对问题二的二轮投票模型进行多次模拟验证，输出报错统计信息

7,**task3_sample.cpp**：对问题三中题目给出的初始情况使用我们的模型进行方案求解，输出方案，调整结果和累积误差

8, **方案表**：问题三题目给出的初始情况的具体方案

9,**task3_sim.cpp**：对问题三的模型进行多次模拟验证，输出累积误差和统计信息。

1,**readme**：

```
支撑材料说明

task1文件夹下问题1第一小问的相关数据：
    data1–3为task1.cpp运行三次分别输出的结果，除最后两行外，每两行表示一次报错的具体数据。
        每两行中，第一行4个正整数数，分别表示两个发射源编号，接收点编号，发射源之间间隔（1表示间隔40度，2表示间隔80度……）
        第二行两个实数，表示需要定位的接收点的带误差的真实位置

task2文件夹下问题1第二小问的相关数据：
    task2_simple_data为task2_simple.cpp的输出结果，每行表示在某个误差范围内，随机模拟1000000次角度区间定位模型后，报错的统计信息
        每行第一个数表示误差范围，第二个数表示报错数据中误差最大的距离，第三个数表示第二类错误（即接收点定位错误）的频率，第四个数表示第一类错误（即发射点编号判断错误）的频率。
    task2_vote_data为task2_vote.cpp的输出结果，每行表示在某个误差范围内，随机模拟1000000次投票机制模型后，报错的统计信息
        每行第一个数表示误差范围，第二个数表示报错数据中误差最大的距离，第三个数表示第二类错误（即接收点定位错误，判断方法为真实坐标和计算结果坐标作差）的频率，第四个数表示第一类错误（即发射点编号判断错误）的频率，第五个数表示通过角度信息判断定位错误的方法计算得到的错误率
    task2_vote2_data为task2_vote2.cpp的输出结果，每行表示在某个误差范围内，随机模拟1000000次二轮投票模型后，报错的统计信息
        每行第一个数表示误差范围，第二个数表示报错数据中误差最大的距离，第三个数表示第二类错误（即接收点定位错误，判断方法为真实坐标和计算结果坐标作差）的频率，第四个数表示第一类错误（即发射点编号判断错误）的频率，第五个数表示通过角度信息判断定位错误的方法计算得到的错误率
    task2_vote_data和task2_vote2_data中每行第3个数和第5个数都相同，可以证明论文中提到的用角度判断定位准确度的方法的正确性

task3文件夹下问题1第三小问的相关数据：
    task3_sample_data为题中给出的样例初始结果的调整方案和调整结果，是task3_sample.cpp的输出结果
        前253行为具体方案，以ask开头的一行是一次查询操作，ask后的数字表示此次查询选择的发射点编号；以move开头的一行是一次移动操作，每行第一个整数表示移动的无人机编号，接下来两个实数表示移动的(dx, dy)
        接下来18行，每两行表示一家无人机的理想位置和调整后的位置
        最后一行表示所有无人机的偏差距离和（即累计误差）
    task3_sim为进一步验证模型准确性和求解稳定性的结果，是task3_sim.cpp的输出结果
        每行一个数表示一次模拟的累计误差，最后一行表示累积误差超过0.001的模拟次数（为0表示累积误差均控制在 0.001 以内）
```

2,**geometry.h**：

```cpp
#include<bits/stdc++.h>
using namespace std;

const double pi=acos(-1);
const int N=20;
const double eps=1e-8;
const double INF=1e16;
double radius=100;
struct point
{
    double x,y;
    void out()
    {
        printf("%.8lf %.8lf\n",x,y);
    }
};
point operator +(point a,point b){return (point){a.x+b.x,a.y+b.y};}
point operator -(point a,point b){return (point){a.x-b.x,a.y-b.y};}
point operator *(point a,double b){return (point){a.x*b,a.y*b};}
point operator /(point a,double b){return (point){a.x/b,a.y/b};}
double cross(point a,point b){return a.x*b.y-a.y*b.x;}
double dot(point a,point b){return a.x*b.x+a.y*b.y;}
double len(point a){return sqrt(a.x*a.x+a.y*a.y);}
double angle(point a,point b1,point b2)
{
    double A=len(b1-b2),B=len(a-b1),C=len(a-b2);
    return acos((B*B+C*C-A*A)/(2*B*C));
}
point get_xy(double r,double theta)
{
    return (point){r*cos(theta),r*sin(theta)};
}
double rand_double(){return ((double)rand()/RAND_MAX-0.5)*2;}
void ask(int p0,int p1,int p2,int p3)
{
    cout<<"ask "<<p0<<' '<<p1<<' '<<p2<<' '<<p3<<endl;
}
void ask(int p0,int p1,int p2)
{
    cout<<"ask "<<p0<<' '<<p1<<' '<<p2<<endl;
}
void ask(int p0,int p1)
{
    cout<<"ask "<<p0<<' '<<p1<<endl;
}
point add_ran(point now)
{
    double radius=len(now);
    double ang=atan2(now.y,now.x);
    radius+=rand_double()*5*3;
    ang+=rand_double()*0.052*3;
    return get_xy(radius,ang);
}
point add_ran(point now,double radio)
{
    double radius=len(now);
    double ang=atan2(now.y,now.x);
    radius+=rand_double()*5*radio/10;
    ang+=rand_double()*0.052*radio/10;
```

```
    return get_xy(radius,ang);
}
point add_ran_on_circle_edge(point now,double radio)
{
    point delta=get_xy(radio,rand_double()*pi);
    return now+delta;
}
double Atan2(double y,double x)
{
    if(y<0)return atan2(-y,-x);
    else return atan2(y,x);
}
pair<double,double> get_rt(int b,int c,int d,double a1,double a2)
{
    double a3=(double)abs(b-c)*40/360*2*pi;
    if(a3>pi)a3=2*pi-a3;
    double t0=a3-a1-a2;
    double t1[5];
     t1[1]=Atan2(sin(t0),sin(a2)/sin(a1)+cos(t0));//-
     t1[2]=Atan2(-sin(a1+a2+a3),sin(a2)/sin(a1)+cos(a1+a2+a3));//+

    t0=a2-a1-a3;
     t1[3]=Atan2(sin(t0),cos(t0)-sin(a2)/sin(a1));//+
    t0=a1-a2-a3;
     t1[4]=Atan2(sin(t0),sin(a2)/sin(a1)-cos(t0));//-

    point ans[5];
    for(int i=1;i<=4;i++)
    {
        double ra=radius*(sin(t1[i])/sin(a1)),td=40.0*(b-1)/360*2*pi+(pi-t1[i]-a1)*((i==1||i==4)
    ?(-1):1);
        ans[i]=get_xy(ra,td);
    }

    point std_ans=get_xy(radius,(d-1)*40.0/360*2*pi);

    pair<double,int> tmp[5];
    for(int i=1;i<=4;i++)tmp[i]=make_pair(len(std_ans-ans[i]),i);
    sort(tmp+1,tmp+5);
    int i=tmp[1].second;
    double ra=radius*(sin(t1[i])/sin(a1)),td=40.0*(b-1)/360*2*pi+(pi-t1[i]-a1)*((i==1||i==4)?(-1):1)
    ;
    if(td>pi*2)td-=pi*2;
    if(td<0)td+=pi*2;
    return make_pair(ra,td);
}
```

3,task1.cpp:

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#include "geometry.h"
using namespace std;
point p[N];
int id[N];

int sum[5];
const int T=10000000;
int main()
{
```

```
    srand(time(0));
    for(int i=1;i<=9;i++)p[i]=get_xy(radius,40.0*(i-1)/360*2*pi),id[i]=i;
    // freopen("data3","w",stdout);
    double mx=0;
    for(int i=1;i<=T;i++)
    {
        random_shuffle(id+1,id+1+9);
        int b=id[1],c=id[2],d=id[3],a=0;
        if(cross(p[b],p[c])<0)swap(b,c);
        point fakeD=add_ran(p[d]);
        double a1=angle(fakeD,p[b],p[a]);
        double a2=angle(fakeD,p[c],p[a]);

        //p[d].out();
        //fakeD.out();
        //cout<<b<<' '<<c<<' '<<d<<endl;
        //cout<<len(fakeD)<<' '<<atan2(fakeD.y,fakeD.x)+pi*2*(atan2(fakeD.y,fakeD.x)<0?1:0)<<endl;
        pair<double,double> ans=get_rt(b,c,d,a1,a2);
        //cout<<ans.first<<' '<<ans.second<<endl;
        point q=(point){ans.first*cos(ans.second),ans.first*sin(ans.second)};
        double err=len(q-fakeD);
        mx=max(err,mx);
        if(err>eps)
        {
            cout<<b<<' '<<c<<' '<<d<<' ';
            int t=abs(b-c);
            if(t>4)t=9-t;
            cout<<t<<endl;sum[t]++;
            fakeD.out();
        }
    }
    printf("%d %d %d %d\n",sum[1],sum[2],sum[3],sum[4]);
    cout<<mx<<endl;
    return 0;
}
```

4,task2_simple.cpp:

```
#pragma GCC optimize(3)
#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#include<bits/stdc++.h>
#include "geometry.h"
using namespace std;

point s[N];
const int n=9;
const int T=1000000;
int id[N];

int tot,tot_wrongc;
pair<double,double> solve2(int num,double ang1,double ang2,double ang3,int trueC)
{
    double per_para=20.0/360*2*pi;
    int para=round(ang1/per_para);
    if(para==0)para++;
    int c1=para+1,c2=10-para;

    double t1=(double)abs(num-c1)*40/360*2*pi;
```

```
    if(t1>pi)t1=2*pi-t1;
    double t2=(double)abs(num-c2)*40/360*2*pi;
    if(t2>pi)t2=2*pi-t2;

    t1=(pi-t1)/2;t2=(pi-t2)/2;
    int C=(fabs(t1-ang3)<fabs(t2-ang3))?c1:c2;
    tot_wrongc+=C!=trueC;
    int B=1;
    if(cross(s[B],s[C])<0)swap(B,C),swap(ang2,ang3);
    return get_rt(B,C,num,ang2,ang3);
}
double mx=0;
int main()
{
    srand(233);
    freopen("task2_simple_data","w",stdout);
    for(int i=1;i<=n;i++)
        s[i]=get_xy(radius,(double)(i-1)/n*2*pi),id[i]=i;
    for(double range=10;range<=40.05;range+=0.1)
    {
        mx=0,tot=0;tot_wrongc=0;
        for(int i=1;i<=T;i++)
        {
            random_shuffle(id+2,id+10);
            int b=1,c=id[2],d=id[3];
            point fakeD=add_ran(s[d],range);

            double ang1=angle(fakeD,s[c],s[b]);
            double ang2=angle(fakeD,s[0],s[b]);
            double ang3=angle(fakeD,s[0],s[c]);

            pair<double,double> ans=solve2(d,ang1,ang2,ang3,c);
            //fakeD.out();
            point ansXY=get_xy(ans.first,ans.second);
            //ansXY.out();
            //cout<<len(fakeD-ansXY)<<endl;
            tot+=len(fakeD-ansXY)>eps;
            mx=max(mx,len(fakeD-ansXY));
        }
        cerr<<range<<endl;
        cout<<range<<' '<<mx<<' '<<(double)tot/T<<' '<<(double)tot_wrongc/T<<endl;
    }

    return 0;
}
```

5,task2_vote.cpp:

```
#pragma GCC optimize(3)
#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#include<bits/stdc++.h>
#include "geometry.h"
using namespace std;

point s[N];
const int n=9;
const int T1=1000;
const int T2=100000;
```

```
13
14 int id[N];
15
16 int tot,tot_wrongc,tot_wrong_verifiable;
17 double seg[1000][10][10][2][2];
18 double mx=0;
19 int totR;
20 double in(double l,double r,double x){return l-eps<=x&&x<=r+eps;}
21 pair<double,double> solve2_vote(int num,double ang1,double ang2,double ang3,int trueC)
22 {
23     int sum[10];
24     memset(sum,0,sizeof(sum));
25     for(int i=1;i<=totR;i++)
26     {
27         for(int j=2;j<=9;j++)
28         if(j!=num)
29         {
30             if(in(seg[i][num][j][0][0],seg[i][num][j][0][1],ang1)&&in(seg[i][num][j][1][0],seg[i][
    num][j][1][1],ang3))sum[j]++;
31         }
32     }
33     vector<pair<int,int> > tmp;
34     for(int j=2;j<=9;j++)if(j!=num)tmp.push_back(make_pair(-sum[j],j));
35     sort(tmp.begin(),tmp.end());
36
37     int C=tmp[0].second;
38     tot_wrongc+=C!=trueC;
39     int B=1;
40     if(cross(s[B],s[C])<0)swap(B,C),swap(ang2,ang3);
41
42     return get_rt(B,C,num,ang2,ang3);
43 }
44
45 int main()
46 {
47     srand(233);
48     freopen("task2_vote_data","w",stdout);
49     for(int i=1;i<=n;i++)
50         s[i]=get_xy(radius,(double)(i-1)/n*2*pi),id[i]=i;
51     for(double range=0.1;range<=40+eps;range+=0.1)
52     {
53         totR++;
54         for(int rec=2;rec<=9;rec++)
55         {
56             for(int se=2;se<=9;se++)
57             if(se!=rec)
58             {
59                 seg[totR][rec][se][0][0]=INF;
60                 seg[totR][rec][se][0][1]=-INF;
61                 seg[totR][rec][se][1][0]=INF;
62                 seg[totR][rec][se][1][1]=-INF;
63
64                 for(int i=1;i<=T1;i++)
65                 {
66                     point fake_rec=add_ran_on_circle_edge(s[rec],range);
67                     seg[totR][rec][se][0][0]=min(seg[totR][rec][se][0][0],angle(fake_rec,s[1],s[se])
    );
68                     seg[totR][rec][se][0][1]=max(seg[totR][rec][se][0][1],angle(fake_rec,s[1],s[se])
    );
```

```
                    seg[totR][rec][se][1][0]=min(seg[totR][rec][se][1][0],angle(fake_rec,s[0],s[se]));
                    seg[totR][rec][se][1][1]=max(seg[totR][rec][se][1][1],angle(fake_rec,s[0],s[se]));

                }
                // cout<<seg[totR][rec][se][0][0]<<' '<<seg[totR][rec][se][0][1]<<' '<<seg[totR][rec][se][1][0]<<' '<<seg[totR][rec][se][1][1]<<endl;
            }
        }
    }
    for(double range=10;range<=40.05;range+=0.1)
    {
        mx=0,tot=0;tot_wrongc=0;tot_wrong_verifiable=0;
        for(int i=1;i<=T2;i++)
        {
            random_shuffle(id+2,id+10);
            int b=1,c=id[2],d=id[3];
            point fakeD=add_ran(s[d],range);

            double ang1=angle(fakeD,s[c],s[b]);
            double ang2=angle(fakeD,s[0],s[b]);
            double ang3=angle(fakeD,s[0],s[c]);

            pair<double,double> ans=solve2_vote(d,ang1,ang2,ang3,c);
            point ansXY=get_xy(ans.first,ans.second);
            point err=(point){ang1-angle(ansXY,s[c],s[b]),ang2-angle(ansXY,s[0],s[b])};
            tot_wrong_verifiable+=len(err)>eps;
            tot+=len(fakeD-ansXY)>eps;
            mx=max(mx,len(fakeD-ansXY));
        }
        cout<<range<<' '<<mx<<' '<<(double)tot/T2<<' '<<(double)tot_wrongc/T2<<' '<<(double)tot_wrong_verifiable/T2<<endl;
    }
    return 0;
}
```

6,task2_vote2.cpp:

```
#pragma GCC optimize(3)
#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#include<bits/stdc++.h>
#include "geometry.h"
using namespace std;

point s[N];
const int n=9;
const int T1=1000;
const int T2=100000;

int id[N];

int tot,tot_wrongc,tot_wrong_verifiable;
double seg[1000][10][10][2][2];
double mx=0;
int totR;
double in(double l,double r,double x){return l-eps<=x&&x<=r+eps;}
pair<double,double> solve2_vote(int num,double ang1,double ang2,double ang3,int trueC)
```

```
{
    int sum[10];
    memset(sum,0,sizeof(sum));
    for(int i=1;i<=totR;i++)
    {
        for(int j=2;j<=9;j++)
        if(j!=num)
        {
            if(in(seg[i][num][j][0][0],seg[i][num][j][0][1],ang1)&&in(seg[i][num][j][1][0],seg[i][num][j][1][1],ang3))sum[j]++;
        }
    }
    vector<pair<int,int> > tmp;
    for(int j=2;j<=9;j++)if(j!=num)tmp.push_back(make_pair(-sum[j],j));
    sort(tmp.begin(),tmp.end());

    int C=tmp[0].second;
    tot_wrongc+=C!=trueC;
    int B=1;
    if(cross(s[B],s[C])<0)swap(B,C),swap(ang2,ang3);

    return get_rt(B,C,num,ang2,ang3);
}
pair<double,double> solve2_vote2(int num,double ang1,double ang2,double ang3,int trueC,double ang1_2,double ang2_2,double ang3_2,int trueC_2)
{
    int sum[10][10];
    memset(sum,0,sizeof(sum));
    for(int i=1;i<=totR;i++)
    {
        for(int j=2;j<=9;j++)
        for(int j2=2;j2<=9;j2++)
        if(j!=num&&j2!=num&&j!=j2)
        {
            if(in(seg[i][num][j][0][0],seg[i][num][j][0][1],ang1)&&in(seg[i][num][j][1][0],seg[i][num][j][1][1],ang3)
            &&in(seg[i][num][j2][0][0],seg[i][num][j2][0][1],ang1_2)&&in(seg[i][num][j2][1][0],seg[i][num][j2][1][1],ang3_2))sum[j][j2]++;
        }
    }
    vector<pair<int,pair<int,int> > > tmp;
    for(int j=2;j<=9;j++)
        for(int j2=2;j2<=9;j2++)
        if(j!=num&&j2!=num&&j!=j2)
            tmp.push_back(make_pair(-sum[j][j2],make_pair(j,j2)));
    sort(tmp.begin(),tmp.end());

    int C1=tmp[0].second.first,C2=tmp[0].second.second;
    tot_wrongc+=(C1!=trueC)||(C2!=trueC_2);
    int B=1;
    if(cross(s[B],s[C2])<0)swap(B,C2),swap(ang2_2,ang3_2);

    return get_rt(B,C2,num,ang2_2,ang3_2);
}

int main()
{
    srand(233);
    freopen("task2_vote2_data","w",stdout);
```

```
77      for(int i=1;i<=n;i++)
78          s[i]=get_xy(radius,(double)(i-1)/n*2*pi),id[i]=i;
79      for(double range=0.1;range<=40+eps;range+=0.1)
80      {
81          totR++;
82          for(int rec=2;rec<=9;rec++)
83          {
84              for(int se=2;se<=9;se++)
85              if(se!=rec)
86              {
87                  seg[totR][rec][se][0][0]=INF;
88                  seg[totR][rec][se][0][1]=-INF;
89                  seg[totR][rec][se][1][0]=INF;
90                  seg[totR][rec][se][1][1]=-INF;
91
92                  for(int i=1;i<=T1;i++)
93                  {
94                      point fake_rec=add_ran_on_circle_edge(s[rec],range);
95                      seg[totR][rec][se][0][0]=min(seg[totR][rec][se][0][0],angle(fake_rec,s[1],s[se]));
96                      seg[totR][rec][se][0][1]=max(seg[totR][rec][se][0][1],angle(fake_rec,s[1],s[se]));
97                      seg[totR][rec][se][1][0]=min(seg[totR][rec][se][1][0],angle(fake_rec,s[0],s[se]));
98                      seg[totR][rec][se][1][1]=max(seg[totR][rec][se][1][1],angle(fake_rec,s[0],s[se]));
99
100                 }
101                 // cout<<seg[totR][rec][se][0][0]<<' '<<seg[totR][rec][se][0][1]<<' '<<seg[totR][rec][se][1][0]<<' '<<seg[totR][rec][se][1][1]<<endl;
102             }
103         }
104     }
105     for(double range=10;range<=40.05;range+=0.1)
106     {
107         mx=0,tot=0;tot_wrongc=0;tot_wrong_verifiable=0;
108         for(int i=1;i<=T2;i++)
109         {
110             random_shuffle(id+2,id+10);
111             int b=1,c=id[2],d=id[3];
112             point fakeD=add_ran(s[d],range);
113
114             double ang1=angle(fakeD,s[c],s[b]);
115             double ang2=angle(fakeD,s[0],s[b]);
116             double ang3=angle(fakeD,s[0],s[c]);
117
118             pair<double,double> ans=solve2_vote(d,ang1,ang2,ang3,c);
119             point ansXY=get_xy(ans.first,ans.second);
120             point err=(point){ang1-angle(ansXY,s[c],s[b]),ang2-angle(ansXY,s[0],s[b])};
121
122             int c2=id[4];
123             double ang1_2=angle(fakeD,s[c2],s[b]);
124             double ang2_2=angle(fakeD,s[0],s[b]);
125             double ang3_2=angle(fakeD,s[0],s[c2]);
126             if(len(err)>eps)ans=solve2_vote2(d,ang1,ang2,ang3,c,ang1_2,ang2_2,ang3_2,c2),tot_wrongc--;
127             ansXY=get_xy(ans.first,ans.second);
128             err=(point){ang1-angle(ansXY,s[c],s[b]),ang2-angle(ansXY,s[0],s[b])};
129
```

```
            tot_wrong_verifiable+=len(err)>eps;
            tot+=len(fakeD-ansXY)>eps;
            mx=max(mx,len(fakeD-ansXY));
        }
        cerr<<range<<endl;
        cout<<range<<' '<<mx<<' '<<(double)tot/T2<<' '<<(double)tot_wrongc/T2<<' '<<(double)
    tot_wrong_verifiable/T2<<endl;
    }
    return 0;
}
```

7,task3_sample.cpp:

```
#include<bits/stdc++.h>
using namespace std;
#include "geometry.h"

point p[N],s[N],la[N];
double th[N],ri[N];
const double sample
    [][2]={{0,0},{100,0},{98,40.10},{112,80.21},{105,119.75},{98,159.86},{112,199.96},{105,240.07},{98,280.17},{112,

double ang[N];
void move(int p,point del)
{
    point now=del-la[p];
    la[p]=del;
    printf("move %d %lf %lf\n",p,now.x,now.y);
}
int main()
{
    freopen("task3_sample_data","w",stdout);
    int n=9;double radius=100;
    for(int i=1;i<=n;i++)
    {
        s[i]=get_xy(radius,(double)(i-1)/n*2*pi);
        p[i]=get_xy(sample[i][0],sample[i][1]/360*2*pi);
        la[i]=(point){0,0};
    }

    ask(0,1);
    ask(0,2,3,4);
    ask(0,5,6,7);
    ask(0,8,9);
    double L[N],R[N];
    for(int i=2;i<=9;i++)
    {
        L[i]=(double)40*(i-1)/360*2*pi,R[i]=L[i];
        L[i]-=2.0*pi*20/360,R[i]+=2.0*pi*20/360;
        ang[i]=pi-angle(p[1],p[0],p[i])-angle(p[i],p[0],p[1]);
    }
    for(int T=1;T<=20;T++)
    {
        for(int i=2;i<=9;i++)
        if(R[i]-L[i]>eps)
        {
            double mid=(L[i]+R[i])/2;
            point delta=(point){2*cos(mid),2*sin(mid)};delta=delta*0.5*(1);
            move(i,delta);
            delta=delta+p[i];
```

```
                double tmp_ang=pi-angle(p[1],p[0],delta)-angle(delta,p[0],p[1]);
                if(i<=5)if(tmp_ang>ang[i])R[i]=mid;else L[i]=mid;
                else if(tmp_ang<ang[i])R[i]=mid;else L[i]=mid;
            }
            ask(0,1);
            ask(0,2,3,4);
            ask(0,5,6,7);
            ask(0,8,9);
        }
        L[1]=R[1]=(L[2]+R[2])/2-ang[2];

        for(int i=1;i<=9;i++)
        {
            th[i]=(L[i]+R[i])/2;
            int ne=i+1;if(ne==10)ne=1;
            double db=i!=1,da=ne!=1;
            double alpha1=angle(p[i],p[0],p[ne]),alpha2=angle(p[i]+la[i],p[0],p[ne]+la[ne]);
            double beta1=angle(p[ne],p[0],p[i]),beta2=angle(p[ne]+la[ne],p[0],p[i]+la[i]);
            double b=(sin(alpha2)/sin(beta2)*db-da)/(sin(alpha1)/sin(beta1)-sin(alpha2)/sin(beta2));
            ri[i]=b;
        }
        for(int i=1;i<=9;i++)move(i,get_xy(ri[i],th[i])-s[i]);
        double sum_err=0;
        for(int i=1;i<=9;i++)s[i].out(),(p[i]-la[i]).out(),sum_err+=len(s[i]-p[i]+la[i]);
        cout<<sum_err<<endl;
        return 0;
}

/*
103.407 -0.585212
104.088 40.2568
97.8435 77.3175
103.147 120.048
102.273 160.142
99.78 -160.74
100.568 -120.705
96.825 -76.9355
99.6751 -38.5416
40.10009766 0.00009766 12
80.21240234 0.00240234 12
119.74853516 -0.00146484 12
159.86083984 0.00083984 12
199.95849609 -0.00150391 12
240.07080078 0.00080078 12
280.16845703 -0.00154297 12
320.28076172 0.00076172 12
*/
```

8, 方案表

1	2	3
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 0.766044 0.642788	move 2 -0.123257 0.123257	move 2 0.064319 -0.058938
move 3 0.173648 0.984808	move 3 -0.173648 0.015192	move 3 0.087156 -0.003805
move 4 -0.500000 0.866025	move 4 0.157980 0.073667	move 4 -0.080598 -0.033385
move 5 -0.939693 0.342020	move 5 0.073667 0.157980	move 5 -0.040282 -0.077382
move 6 -0.939693 -0.342020	move 6 -0.045115 0.168372	move 6 0.018882 -0.085171
move 7 -0.500000 -0.866025	move 7 0.157980 -0.073667	move 7 -0.080598 0.033385
move 8 0.173648 -0.984808	move 8 0.168372 0.045115	move 8 -0.083201 -0.026233
move 9 0.766044 -0.642788	move 9 0.099981 0.142788	move 9 -0.046873 -0.073576
4	5	6
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 0.030171 -0.031517	move 2 0.014562 -0.016244	move 2 0.007147 -0.008240
move 3 0.043370 -0.004750	move 3 0.021597 -0.003083	move 3 0.010772 -0.001718
move 4 -0.039130 -0.019297	move 4 -0.019240 -0.010284	move 4 -0.009535 -0.005299
move 5 -0.017572 -0.039935	move 5 -0.008128 -0.020245	move 5 -0.003898 -0.010188
move 6 0.012209 -0.041887	move 6 0.006787 -0.020734	move 6 0.003563 -0.010310
move 7 -0.039130 0.019297	move 7 -0.019240 0.010284	move 7 -0.009535 0.005299
move 8 -0.042379 -0.010370	move 8 -0.021349 -0.004489	move 8 -0.010710 -0.002070
move 9 -0.025799 -0.035185	move 9 -0.013469 -0.017162	move 9 -0.006874 -0.008470
7	8	9
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 0.003540 -0.004149	move 2 0.001761 -0.002082	move 2 0.000879 -0.001043
move 3 0.005379 -0.000903	move 3 0.002688 -0.000463	move 3 -0.001344 0.000232
move 4 -0.004746 -0.002688	move 4 -0.002367 -0.001354	move 4 0.001183 0.000678
move 5 -0.001907 -0.005110	move 5 -0.000943 -0.002559	move 5 -0.000469 -0.001280
move 6 0.001823 -0.005140	move 6 0.000922 -0.002566	move 6 0.000464 -0.001282
move 7 -0.004746 0.002688	move 7 -0.002367 0.001354	move 7 -0.001182 0.000679
move 8 -0.005363 -0.000991	move 8 -0.002684 -0.000485	move 8 0.001342 0.000241
move 9 -0.003472 -0.004207	move 9 -0.001744 -0.002096	move 9 0.000873 0.001048

10	11	12
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 -0.000439 0.000522	move 2 0.000220 -0.000261	move 2 -0.000110 0.000130
move 3 0.000672 -0.000116	move 3 -0.000336 0.000058	move 3 0.000168 -0.000029
move 4 0.000592 0.000338	move 4 -0.000296 -0.000169	move 4 -0.000148 -0.000085
move 5 0.000234 0.000640	move 5 0.000117 0.000320	move 5 0.000059 0.000160
move 6 0.000233 -0.000641	move 6 -0.000116 0.000320	move 6 0.000058 -0.000160
move 7 -0.000591 0.000340	move 7 0.000295 -0.000170	move 7 0.000148 -0.000085
move 8 -0.000671 -0.000121	move 8 -0.000336 -0.000060	move 8 -0.000168 -0.000030
move 9 0.000436 0.000524	move 9 0.000218 0.000262	move 9 -0.000109 -0.000131
13	14	15
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 0.000055 -0.000065	move 2 0.000027 -0.000033	move 2 0.000014 -0.000016
move 3 -0.000084 0.000014	move 3 -0.000042 0.000007	move 3 0.000021 -0.000004
move 4 0.000074 0.000042	move 4 0.000037 0.000021	move 4 -0.000018 -0.000011
move 5 -0.000029 -0.000080	move 5 -0.000015 -0.000040	move 5 0.000007 0.000020
move 6 0.000029 -0.000080	move 6 0.000015 -0.000040	move 6 0.000007 -0.000020
move 7 0.000074 -0.000043	move 7 -0.000037 0.000021	move 7 -0.000018 0.000011
move 8 0.000084 0.000015	move 8 -0.000042 -0.000008	move 8 0.000021 0.000004
move 9 -0.000054 -0.000066	move 9 0.000027 0.000033	move 9 -0.000014 -0.000016
16	17	18
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 -0.000007 0.000008	move 2 -0.000003 0.000004	move 2 -0.000002 0.000002
move 3 0.000010 -0.000002	move 3 0.000005 -0.000001	move 3 0.000003 -0.000000
move 4 -0.000009 -0.000005	move 4 0.000005 0.000003	move 4 0.000002 0.000001
move 5 -0.000004 -0.000010	move 5 0.000002 0.000005	move 5 -0.000001 -0.000003
move 6 0.000004 -0.000010	move 6 -0.000002 0.000005	move 6 -0.000001 0.000003
move 7 0.000009 -0.000005	move 7 -0.000005 0.000003	move 7 0.000002 -0.000001
move 8 0.000010 0.000002	move 8 -0.000005 -0.000001	move 8 0.000003 0.000000
move 9 0.000007 0.000008	move 9 -0.000003 -0.000004	move 9 0.000002 0.000002

19	20	21
ask 0 1	ask 0 1	ask 0 1
ask 0 2 3 4	ask 0 2 3 4	ask 0 2 3 4
ask 0 5 6 7	ask 0 5 6 7	ask 0 5 6 7
ask 0 8 9	ask 0 8 9	ask 0 8 9
move 2 -0.000001 0.000001	move 2 0.000000 -0.000001	move 1 0.000000 0.000004
move 3 0.000001 -0.000000	move 3 0.000001 -0.000000	move 2 -2.407071 -1.798766
move 4 -0.000001 -0.000001	move 4 -0.000001 -0.000000	move 3 1.509311 10.902803
move 5 0.000000 0.000001	move 5 0.000000 0.000001	move 4 -1.606522 3.690133
move 6 0.000000 -0.000001	move 6 0.000000 -0.000001	move 5 2.900425 -0.803409
move 7 -0.000001 0.000001	move 7 0.000001 -0.000000	move 6 -10.363095 -3.689399
move 8 -0.000001 -0.000000	move 8 -0.000001 -0.000000	move 7 -1.889894 -3.527582
move 9 0.000001 0.000001	move 9 -0.000000 -0.000001	move 8 -0.237584 3.004821
		move 9 8.774173 -6.654248

9,**task3_sim.cpp:**

```
#include<bits/stdc++.h>
using namespace std;
#include "geometry.h"

point p[N],s[N],la[N];
double th[N],ri[N];
const double sample
    [][2]={{0,0},{100,0},{98,40.10},{112,80.21},{105,119.75},{98,159.86},{112,199.96},{105,240.07},{98,280.17},{112,

double ang[N];
void move(int p,point del)
{
    point now=del-la[p];
    la[p]=del;
}
int main()
{
    freopen("task3_data","w",stdout);
    int tot_err=0;
    for(int cas=1;cas<=1000000;cas++)
    {
        int n=9;double radius=100;
        for(int i=1;i<=n;i++)
        {
            s[i]=get_xy(radius,(double)(i-1)/n*2*pi);
            p[i]=add_ran(s[i],30);
            la[i]=(point){0,0};
        }
        double L[N],R[N];
        for(int i=2;i<=9;i++)
        {
            L[i]=(double)40*(i-1)/360*2*pi,R[i]=L[i];
            L[i]-=2.0*pi*20/360,R[i]+=2.0*pi*20/360;
            ang[i]=pi-angle(p[1],p[0],p[i])-angle(p[i],p[0],p[1]);
        }
        for(int T=1;T<=20;T++)
        {
```

```
            for(int i=2;i<=9;i++)
            if(R[i]-L[i]>eps)
            {
                double mid=(L[i]+R[i])/2;
                point delta=(point){2*cos(mid),2*sin(mid)};delta=delta*0.5*(1);
                move(i,delta);
                delta=delta+p[i];
                double tmp_ang=pi-angle(p[1],p[0],delta)-angle(delta,p[0],p[1]);
                if(i<=5)if(tmp_ang>ang[i])R[i]=mid;else L[i]=mid;
                else if(tmp_ang<ang[i])R[i]=mid;else L[i]=mid;
            }
        }
        L[1]=R[1]=(L[2]+R[2])/2-ang[2];

        double sum_err=0;
        for(int i=1;i<=9;i++)
        {
            th[i]=(L[i]+R[i])/2;
            int ne=i+1;if(ne==10)ne=1;
            double db=i!=1,da=ne!=1;
            double alpha1=angle(p[i],p[0],p[ne]),alpha2=angle(p[i]+la[i],p[0],p[ne]+la[ne]);
            double beta1=angle(p[ne],p[0],p[i]),beta2=angle(p[ne]+la[ne],p[0],p[i]+la[i]);
            double b=(sin(alpha2)/sin(beta2)*db-da)/(sin(alpha1)/sin(beta1)-sin(alpha2)/sin(beta2));
            ri[i]=b;
            sum_err+=len(get_xy(ri[i],th[i])-p[i]);
        }
        for(int i=1;i<=9;i++)move(i,get_xy(ri[i],th[i])-s[i]);
        cout<<sum_err<<endl;
        tot_err+=sum_err>1e-3;
    }
    cout<<tot_err<<endl;

    return 0;
}
```