基于狼群模拟退火模型的无人机纯方向无源定位编队控制

摘要

无人机集群飞行编队过程中，为增加无线电信号的电磁屏蔽性，应当减少信号发射量。为了在这种情况下保证无人机编队队形控制，无源定位方法应运而生。本文面向纯方向无源定位，基于狼群控制思想和模拟退火思想，提出**分组递归狼群模拟退火算法**，分别解决了圆形编队和锥形编队的队形控制问题。

问题一（1）：本文利用几何原理，建立了定位方程，利用 Matlab 编程求得了该方程的数值解，从而在已知编号且位置无误差的圆心及圆周上两个，共三架无人机为信号源，仅接收相对角度信息的情况下，实现接收信号无人机（下称本机）的定位。

问题一（2）：本文推广问题一（1）中的方法，在给定约束条件下，通过假定本机位置偏差服从正态分布，猜测本机及信号源编号，实现仅需额外一架未知编号无人机作为信号源条件下本机的有效定位。

问题一（3）：本文推广问题一（1）中的方法，假定信号源的位置没有偏差，通过定位方程求解本机位置，并以此指导无人机行动。为解决信号源位置偏差问题，我们先后建立单源控制、简单迭代模型，取得了一定效果，但两个模型均存在缺陷。最终，我们受到狼群控制思想和模拟退火过程的启发，在已有模型基础上，提出分组狼群模拟退火算法，有效解决了无人机集群圆形编队无源定位问题。在本题给定的初始条件中，在**15 次迭代**以内完成了**精度为** 10^{-3} 的圆形编队控制。

问题二: 本文推广了分组狼群模拟退火算法，引入递归思想提出递归狼群模拟退火算法，复用问题一中的模型，将大规模锥形编队控制问题转换为多个等效子锥形编队控制问题，为**任意规模**锥形无人机集群无源定位提出了有效解决方案。

本文的优点为：

1）本文提出的分组递归狼群模拟退火算法的方法具有**收敛稳定，收敛速度快，鲁棒性强**等显著优势。

2）本文代码采用 C++ 面向对象编程，可拓展性强；运用 Matlab 数学工具 coder 将 Matlab 代码编译为 cpp 类型文件，进而使控制代码可**直接移植于单片机平台**。

3）本文方法可通过递归复用应用于任意规模的无人机锥形编队队形控制。

关键字：狼群算法　模拟退火　纯方向无源定位　编队控制　图论

一、 问题重述

1.1 问题背景

无人机群编队控制理论在近年来发展迅速。在无人机集群飞行过程中，应当尽量减少发射无线电信号，以避免外界干扰，或者被敌人发现。为了增加无线电信号的电磁屏蔽性，并增加定位效率，无源定位技术应运而生 [1]。本题中，采用纯方向无源定位的方式进行无人机群的编队队形控制。即在一个每个无人机都有固定编号的编队中，有多个无人机主动发射信号，其余无人机被动接受信号，从中提取位置信息，约定被动接受信号的无人机收到的信息是以该无人机为顶点和该无人机与任意两架主动发射信号无人机连线的夹角。

1.2 问题重述

问题一：在一个 10 架无人机编队中，编号为 FY01 FY09 的无人机均匀的分布在一定半径的圆周上，编号为 FY00 的无人机位于圆心。无人机保持约定的高度飞行，认为所有无人机均在同一水平面上。

（1）FY00 与另外两架圆周上的无人机主动发射信号，其余无人机被动接受信号。假设发射信号的无人机位置无偏差且编号已知，接受信号的无人机位置只有略微偏差。建立被动接受信号无人机的定位模型。

（2）FY00、FY01 与编队中若干未知编号的无人机主动发射信号，假设发射信号的无人机位置无偏差，但仅有 FY00、FY01 编号已知，其余未知，接收信号的无人机位置只有略微偏差。建立被动接受信号无人机的定位模型并分析至少需要几架未知编号的无人机发射信号，才能实现无人机的有效定位。

（3）初始时刻，FY00 位于圆心，其它 9 架无人机分布在半径为 100m 的圆周上，各个无人机的位置略有偏差，其初始位置题目已给出。假设每次调整过程中，可选择 FY00 和圆周上最多三架无人机主动发射信号，被动接收无人机根据接收信号调整自身方位，调整时间可忽略不计。建立无人机编队队形控制模型，给出无人机位置调整方案，使得无人机编队经过多次调整后，9 架无人机均匀分布在某个圆周上。

问题二：无人机采用锥形编队，相邻的三个无人机构成边长为 50m 的等边三角形，仍考虑第一问纯方向无源定位的情形，建立无人机编队队形控制模型，给出无人机位置调整方案。

二、 模型的假设

- 无人机视为质点，具有指定方向和距离移动的能力。具体来说，无人机能够规划自己的加速度进而以目标向量移动。
- 初始位置偏离目标位置的分布符合正态分布。
- 无人机具备基本的实时运算能力和数据存储能力。
- 忽略风向、地转偏向力等方向恒定的影响。将微风扰动等不定向影响视为高斯噪声。
- 无人机预先知道自己的编号和自己在队伍中应处于的位置。

三、 符号说明

表 1　符号说明

符号	含义说明	单位
α	方向信息	°
ρ	被动接收信号的无人机极径	m
θ	被动接收信号的无人机极角	°
α_0	目标位置方向信息	°
S_1^2	无人机极径方差	–
S_2^2	相邻两无人机距离方差	–
B	旋转矩阵	–
A_{ij}	狼群结构矩阵	–
E	局部总误差	–
$\vec{t}$	调整向量	–

四、 问题一（1）：模型建立与求解

4.1 模型建立

对于问题 1，可以使用几何方法进行分析处理。假设被动接收信号的无人机（以下简称“本机”）的编号为 FY0X，发射信号的无人机（以下简称“信号源”）为 FY0N 和 FY0M，则可根据几何学原理，对该问题进行分类讨论。首先，根据假设 1，记 FY00，

FY0X，FY0N 和 FY0M 的所在位置为 O、X、N、M 点，做过 OX 的直线 l，若 MN 在 l 的不同侧，记为情况（a），若 MN 在 l 的同侧，记为情况（b），对于 M 或 N 恰好在 l 上的情况，归类为情况（a）。在实际无人机控制中，FY0X 可根据右手螺旋定则，分别判断 FY0M、FY0N 是在 FY00 的顺时针方向或逆时针方向，若两者所在相对于 FY00 顺逆时针方位相同，则属于情况（b），否则属于情况（a）。在（a）情况中，根据本机位置是在 MN 两点间优弧或是劣弧，可以具体分为（a）与（c）两种情况，但两者的区分主要体现在编程计算中，两者数学关系相同，因此只考虑情况（a）。将上述模型转换为几何图形的结果如图1所示，对图1的几何分析如下：

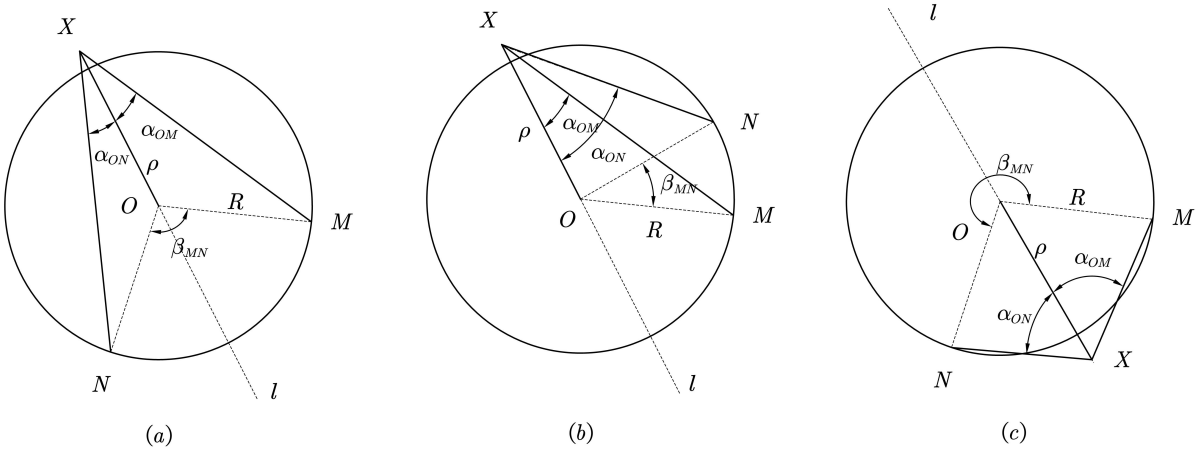


图 1　问题一（1）几何分析

由正弦定理，有

$$\frac{R}{\sin\alpha_{ON}}=\frac{\rho}{\sin N} \tag{1}$$

$$\frac{R}{\sin\alpha_{OM}}=\frac{\rho}{\sin M} \tag{2}$$

对于情况（a），有

$$\beta_{MN}=N+M+\alpha_{ON}+\alpha_{OM} \tag{3}$$

对于情况（b），有

$$\beta_{MN}=N-M+\alpha_{ON}-\alpha_{OM} \tag{4}$$

将(1)、(2)带入(3)或(4)中，将得到关于 ρ 的一元方程：

$$\beta_{MN}=\arcsin\left(\frac{\rho\sin\alpha_{ON}}{R}\right)+\arcsin\left(\frac{\rho\sin\alpha_{OM}}{R}\right)+\alpha_{ON}+\alpha_{OM} \tag{5}$$

或

$$\beta_{MN}=\arcsin\left(\frac{\rho\sin\alpha_{ON}}{R}\right)-\arcsin\left(\frac{\rho\sin\alpha_{OM}}{R}\right)+\alpha_{ON}-\alpha_{OM} \tag{6}$$

由于两个方程没有解析解，运用 Matlab 编程，可求得不同初始状态下的数值解。

4.2 模型求解

在测试模式时，假设该无人机组成的圆的半径为 $100m$，表2是问题一（1）测试样例，其中样例 1，2 是当信号源位于本机两侧的情况，样例 3，4 是当信号源位于本机同一侧的情况：

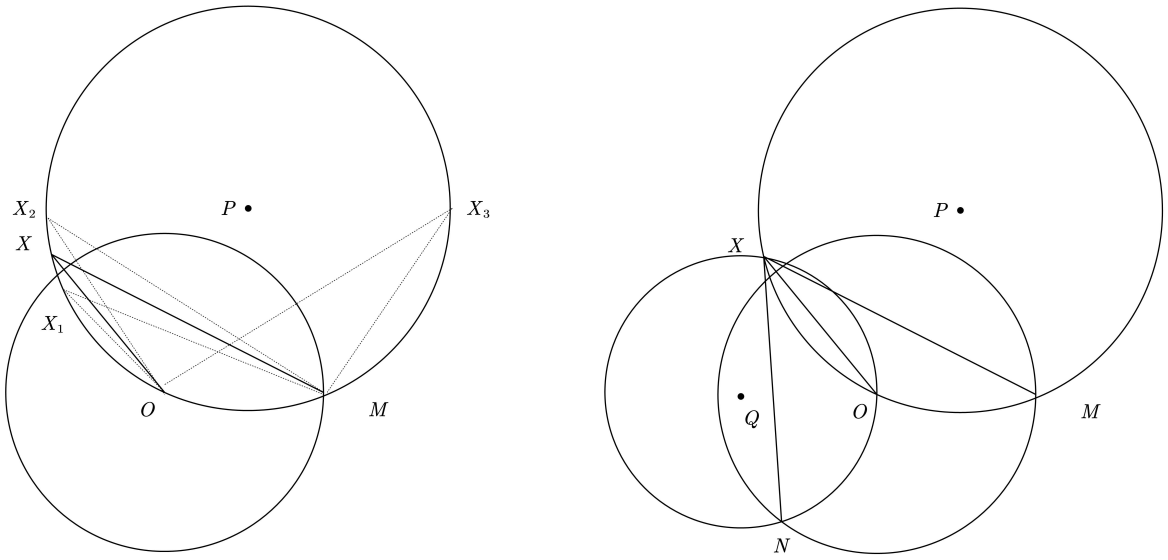
表 2　问题一（1）定位模型测试样例

样例编号	信号源 1 编号	角度 1	信号源 2 编号	角度 2	本机的极径	本机的极角
1	2	$50.00°$	6	$-50.00°$	$100.00m$	$120.00°$
2	2	$49.00°$	6	$-50.00°$	$101.48m$	$121.02°$
3	2	$50.00°$	3	$70.00°$	$100.00m$	$120.00°$
4	2	$51.00°$	3	$70.00°$	$102.00m$	$116.56°$

五、 问题一（2）：模型的建立与求解

5.1 模型建立

首先，根据无人机 FY00 和 FY01 给出的角度 α_{OM}，我们可以以 FY00 与 FY01 的位置为 O 点、M 点为圆上的点、以 α_{OM} 为 OM 所对应圆周角做圆 P，如图2（a）所示，其中圆 O 是无人机目标分布圆周，我们可以发现本机可能在圆 P 的上的任意位置。因此，通过两架无人机是无法对本机进行定位的。因此我们需要另一架发射信号的无人机辅助定位，根据图2（b）可知，若知道圆上两点，M 点、N 点，则可以再以 α_{ON} 为 ON 所对应圆周角做圆 Q，那么圆 P 与圆 Q 的两个交点分别是 O 点与本机位置，即可以通过 α_{OM}、α_{ON} 确定本机位置。



(a) (b)

图 2　问题一（2）两架无人机定位的几何分析

对于圆周上两架无人机，其无人机编号与 α 之间具有表3中的关系。

表 3　无人机编号差值与 $|\alpha_0|$ 之间的关系

无人机编号差值	$\lvert\alpha_0\rvert$
1	70°
2	50°
3	30°
4	10°

由于本机位置只是略有误差，所以根据 α_{OM} 的数值可以大致知道本机的编号。根据另一信号源发射的信号 α_{ON} 可以基本确定发射源 2 的编号。由于在问题一（2）中发射信号的发射源无人机位置无偏差，而本机位置略有偏差，所以发射源 2 存在两种可能。根据假设，无人机的位置偏差服从正态分布，如(7)所示，

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{7}$$

其中 $\mu = \alpha_0 = \pm 70, \pm 50, \pm 30, \pm 10$，根据 3σ 原则，$\sigma = \frac{20}{6}$。我们根据推测的编号分别计算信号源 2 在两个位置的概率密度并归一化，得到信号源 2 在两个位置的概率，再利用问题一（1）中的算法进行计算，得到本机所在位置。

5.2 模型求解

在测试模式时，假设该无人机组成的圆的半径为 $100m$，表4是问题一（2）测试样例，其中样例 1，2 是信号源位于本机两侧的情况，样例 3，4 是信号源位于本机同一侧的情况。我们可以发现在本机位置偏差不大的情况下三架飞机可以实现有效定位。

表 4　问题一（2）定位模型测试样例

样例编号	α_{OM}	α_{ON}	位置 1 概率	位置 1 极径	位置 1 极角	位置 2 概率	位置 2 极径	位置 2 极角
1	$10.00°$	$-50.00°$	1.0000	100.00	160.00	1.523×10^{-8}	33.33	$166.67°$
2	$9.50°$	$-49.50°$	1.0000	102.56	160.75	5.573×10^{-7}	39.11	$66.80°$
3	$10.00°$	$30.00°$	1.0000	100.00	160.00	1.523×10^{-8}	196.96	$150.00°$
4	$9.50°$	$30.50°$	1.0000	91.56	161.80	3.746×10^{-8}	192.37	$151.98°$

六、 问题一（3）:

在问题一（3）中，本机可以通过本机编号获得目标位置相对于 FY00 的极坐标。因此，定义其目标位置向量为表5。在问题一（3）中我们采用先定位再移动的思想调整无人机的位置。首先，假设信号源（包括 FY00 与圆周上任意两架无人机）位置准确，联立式(1)、(2)、(3)或(4)，计算在该条件下，本机相对于 FY00 的极坐标，称该向量为计算位置相量。然后，本机将目标位置向量减去计算位置向量，得出本机调整向量。再根据调整向量的方向和大小，进行位置调整。

表 5　无人机目标位置

无人机编号	极坐标	无人机编号	极坐标
1	$(100, 0°)$	6	$(100, 200°)$
2	$(100, 40°)$	7	$(100, 240°)$
3	$(100, 80°)$	8	$(100, 280°)$
4	$(100, 120°)$	9	$(100, 320°)$
5	$(100, 160°)$		

在问题一（3）中，我们设定了两个评价无人机编队分布精确程度的指标。指标 1，所有无人机极径方差 S_1^2，所有相邻两架无人机距离方差 S_2^2。

$$S_1^2 = \frac{1}{n}\Sigma(\rho_i - \bar{\rho})^2, S_2^2 = \frac{1}{n}\Sigma(x_i - \bar{x})^2$$

在问题一（3）中我们采用面向对象的编程思维，基于 C++ 语言，封装了无人机类，如图3，以及模拟实体类，如图4。考虑到 matlab 在数学求解过程中的功能完善，在问题一（3）中我们沿用了前两问中制作的基于 matlab 的数学工具。因此需要完成 matlab 与 c++ 项目的连接。为了成功将 matlab 编译为 cpp 类型文件，我们调整了 matlab 中使用的非基于 C 语言函数，并对于 matlab 弱类型变量进行重定义，最终利用 matlab 中的 coder 工具箱编译并生成 cpp 类型文件。

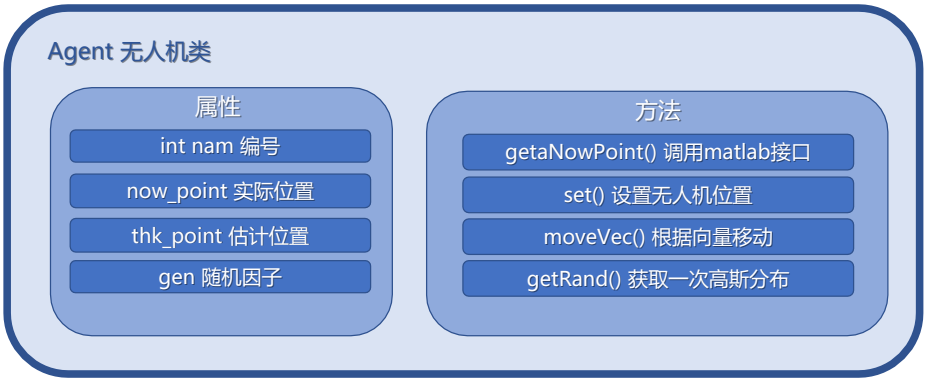


图 3　无人机类

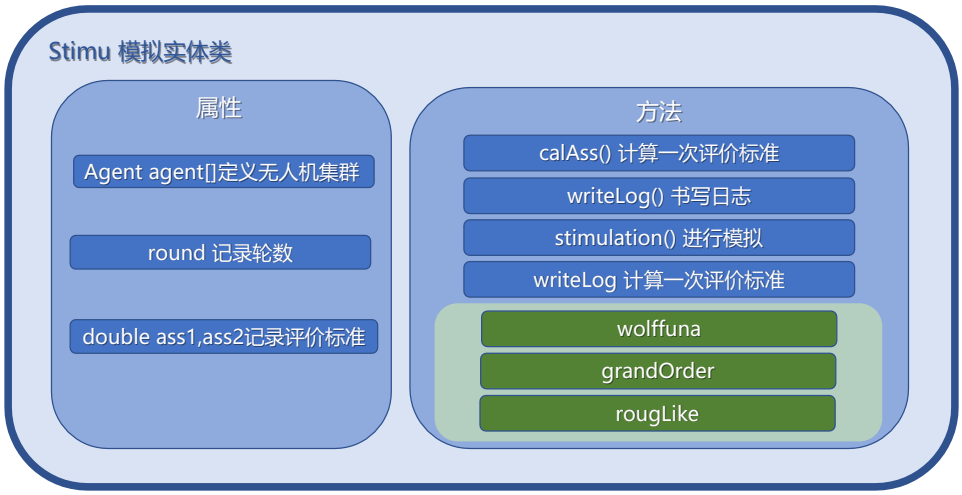


图 4　模拟实体类

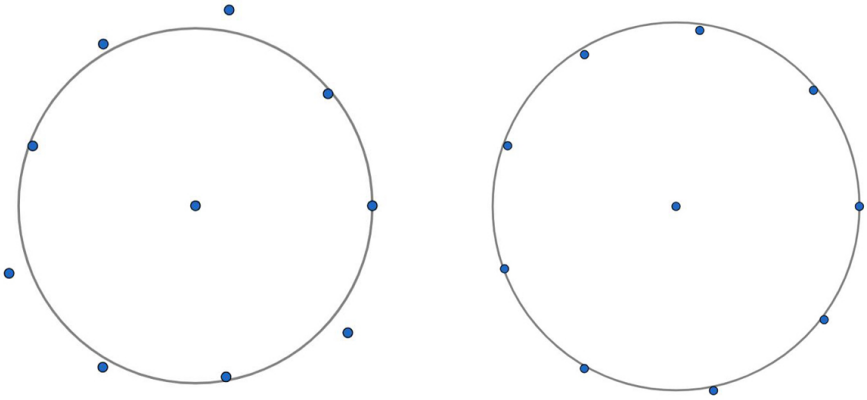
6.1 单源控制

6.1.1 模型建立

在单源控制模型中，我们设定两个无人机为标准无人机 n_1、n_2，其余所有为一般无人机。假设标准无人机位置正确且不进行移动，所有一般无人机均以标准无人机发射信号为基准计算当前位置，并向目标位置移动一次。

6.1.2 模型求解

样例 1 中，我们以 1 号机、2 号机为标准无人机进行移动，通过计算我们发现在移动后中除了 2 号机本身位置存在误差，其余无人机距离在目标位置的偏差大幅减小，如图5其中蓝点为无人机分布、图6。初始时刻，$S_1^2 = 35,14$，$S_2^2 = 5.07$；结束时刻，$S_1^2 = 4.31$，$S_2^2 = 6.93$。由此可知，该方法具有良好的效果。



(a) 初始位置　　(b) 结束位置

图 5　单源控制模型测试样例 1 结果

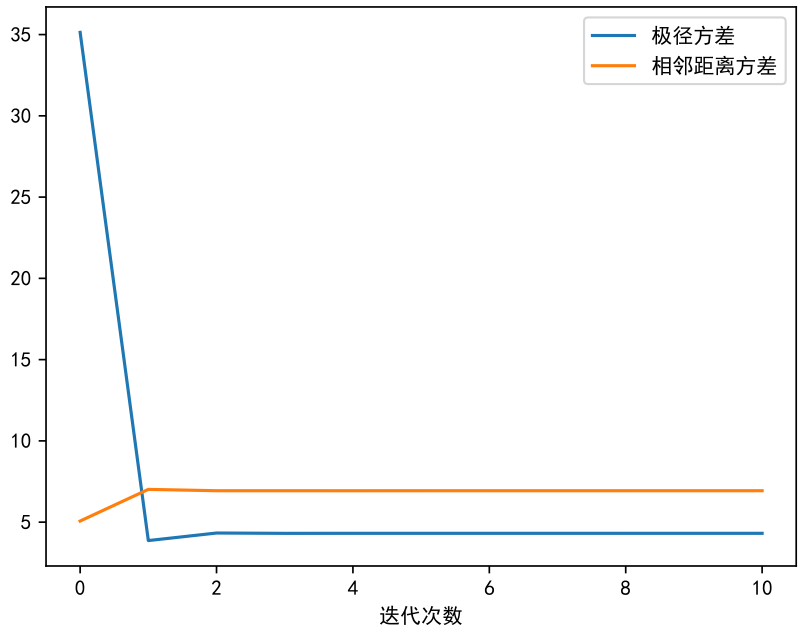
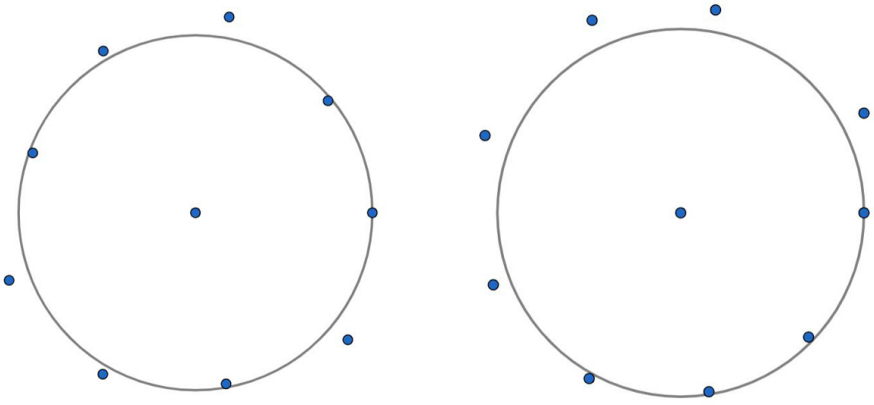


图 6　单源控制模型测试样例 1 误差评估

样例 2 中，我们以 1 号机、2 号机为标准无人机进行移动，通过计算我们发现移动后所有无人机的位置对比于移动之前并没有改善，如图7其中蓝点为无人机分布。初始时刻，$S_1^2 = 35,14$，$S_2^2 = 5.07$；结束时刻，$S_1^2 = 43.73$，$S_2^2 = 128.50$。由此可知，该方法效果较差。



(a) 初始位置　　(b) 结束位置

图 7　单源控制模型测试样例 2 结果

6.1.3 模型评价

单源控制的优点在于迭代次数少，通信需要发射的电磁波信号少，可以尽可能的保持电磁静默。但是，通过样例 1 与样例 2 的测试，我们发现标准无人机的位置会对该方法的计算结果产生较大影响，当标准无人机位置接近目标位置时单源控制效果好，但是当标准无人机距离目标位置较远时单源控制方法效果很差。

6.2 简单迭代

6.2.1 模型准备

在 2004 年，任伟 [2] 等人在动态变化条件下的信息公示交互拓扑结构中，重点研究了在信息单向传输，多代理之间的信息一致性问题，并证明了当有向图具有生成树时，它的谱半径是一个简单的特征值，也就是说改动态智能体可以实现全局一致收敛。在问题一（3）中，无人机编队中的每一架无人机只接收信号源无人机发出的信号并无反馈，实现了信息的单项传输。根据其传输路径我们可以建立起无人机编队的信息有向图。在简单迭代法中我们需要保证在时序控制过程中无人机编队有向图具有生成树。

6.2.2 模型建立

在简单迭代模型中，无人机编队中每一架无人机在调整方法中具有相同的决策权重。在无人机定位过程中，在信号源无人机绝对误差相同的情况下，距离本机更远的信号源无人机位置的相对误差更小，计算获得的计算位置的绝对误差越小，并保证了无人机有向图具有生成树。因此对于每一架本机，都接收距离本机最远两架信号源无人机发射的 α，并按照计算得到本机调整向量进行移动。

6.2.3 模型求解

我们将题目描述的初始位置作为样例 1 测试简单迭代模型。图8是简单模型测试样例 1 误差评估指标曲线，对于无人机极径方差 S_1^2 评估指标，其在第一次迭代时有明显减小，随后在一定范围内波动且具有减小趋势，对于无人机极径方差 S_2^2 评估指标，同样在波动中减小。总体来说，在此样例中迭代 30 次左右基本实现准确定位。

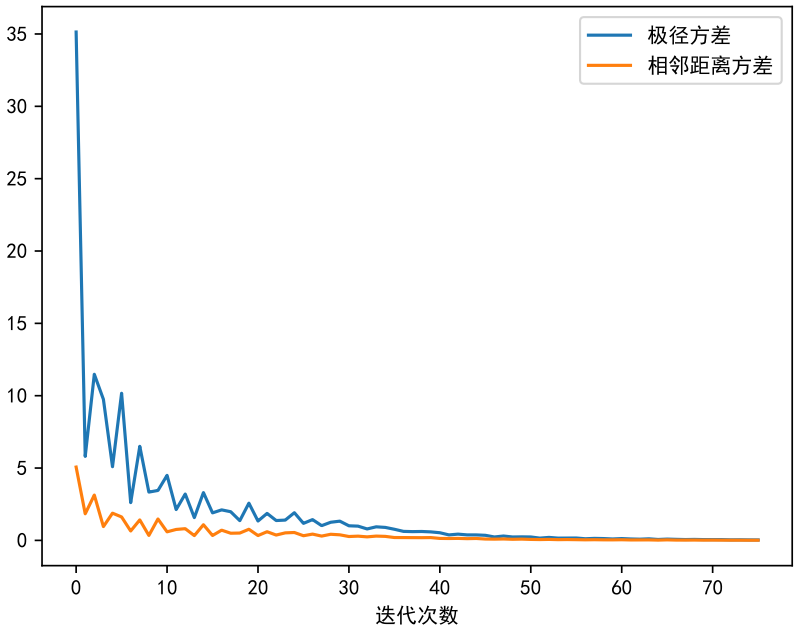


图 8　简单迭代模型测试样例 1 误差评估

考虑到真实情况中无人机的飞行会受到风速、机械结构等因素的干扰。我们在测试样例 1 的基础上增加朝向任意方向与移动距离有关的的高斯白噪声干扰作为测试样例 2。图9是简单模型测试样例 2 误差评估指标曲线，我们可以发现在有干扰的情况下简单迭代模型会趋向发散，最终导致调整失败。

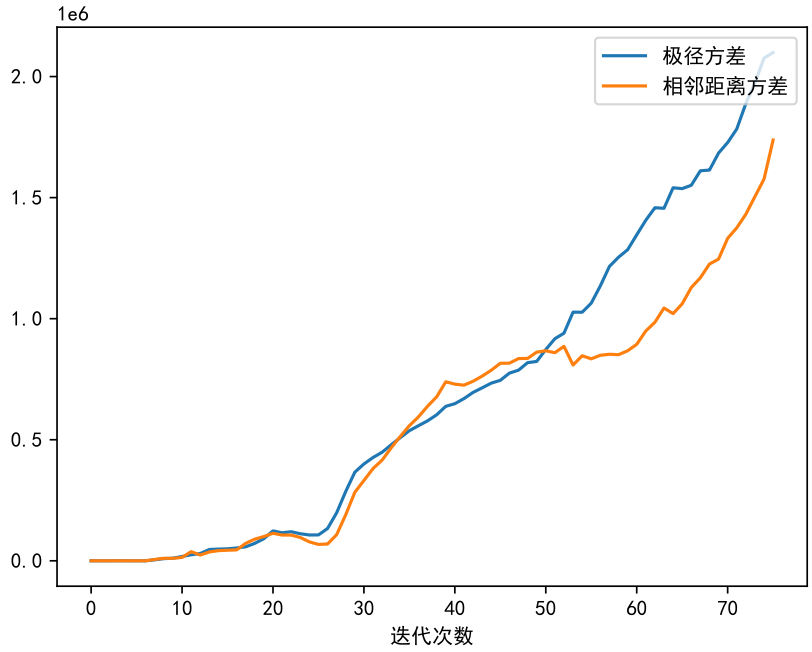


图 9　简单迭代模型测试样例 2 误差评估

6.2.4 模型评价

简单迭代的优点是算法简单，控制效果优越，偏差小，所有周边无人机可以使用同样的控制器，有效减少无人机设计、制作成本。但是，该种方法收敛速度慢，所需迭代次数多，迭代过程中会存在整体偏差反复震荡的过程。并且此种方法对随机扰动的鲁棒性很差。

6.3 分组狼群模拟退火迭代

6.3.1 模型准备

无人机编队飞行的控制思路与自然界社会的社会行为存在相似之处。Chen[3] 等人受此启发，基于狼群社会的管理机制建立了基于图论的狼群层次控制模型，从而提出了一种新的自主编队控制方法。相比于传统控制，狼群层次控制模型设定无人机之间存在上下级关系，层级低的无人机仅受其上级的调配。该方法的显著优势是控制方式简单易行，收敛稳定且速度快。

然而，在 Chen 等人的研究中，单个无人机可以使用自身传感器定位到对方的位置、方向等信息。而对于问题一所采用的无源定位方法，一架无人机至少需要接收三架其他无人机的信号才可以实现自身定位。为此，我们将狼群进行分组，有多只高级狼共同负责低等级狼的定位。同时，由于无源定位中，多个信号源之间不可避免的相对位置误差将严重影响定位效果，若采用严格上下级关系，则将导致误差逐级积累。为此，我们重

新构建了狼群层级，在保持上下级结构的基础上，借鉴“模拟退火”算法，引入带有退火因子 [4] 的收敛系数，使得高等级的狼会先“兼听”低等级狼的位置信息，使随机的误差相互抵消，尽可能的准确定位高等级狼。

为了明确狼群中的层级关系，我们使用无向图表示狼群的分组以及层级关系，记位于第 i 级的编号为 j 的狼为 W_{ij}，定义可以影响到它的狼为狼群结构矩阵 A_{ij}

$$A_{ij} = \begin{bmatrix} i_1 & i_2 & \cdots & i_n \\ j_1 & j_2 & \cdots & j_n \end{bmatrix} \tag{8}$$

向量 r_n 代表第 n 个能影响到 W_{ij} 的狼的信息，其第一行元素为该狼的等级，第二行元素为该狼的编号。r_n 的定义如式(9)所示。

$$r_n = A_{ij} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}, \ a_i = \begin{cases} 0 & i \neq n \\ 1 & i = n \end{cases} \tag{9}$$

为了减少狼群算法中头狼的位置误差带来的影响，我们借鉴模拟退火的思想，我们定义 W_{ij} 听取 r_n 的信息比例为:

$$P = \begin{cases} 1 & i \geq \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot r_n \\ e^{-f(C)} & i < \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot r_n \end{cases} \tag{10}$$

从而使头狼能“兼听”组内下级的信息，利用误差的随机性使之尽可能相互抵消，式(10)中，$f(C)$ 是一个关于信号传递轮数 C 的不小于 0 的非减函数。

6.3.2 模型建立

对于问题一（3），我们将 FY01 至 FY09 分为三组，每组中定义一个组长，在所有无人机中，令 FY00 和各个分组组长为 α 狼（1 级狼），定义其他无人机为 β 狼（2 级狼）。为了尽量减少相对位置带来的误差，我们尽可能让组内成员互相远离，最终，建立的层级关系如图10所示。

在确定位置时，FY04 和 FY07 作为组员，完全听从 FY00 及其组长和另一组内成员提供的位置信息。FY01 作为组长，以式(10)的概率，听取 FY00 及其它组长或 FY00 及本组组员提供的位置信息。由于其它组长能确定其它组组内的无人机位置，这个过程有助于组长帮助组内和组外达成位置一致。

在上述过程中，组长会分别接收来自 FY00 和其它组长以及 FY00 和本组组员的信号。由于两次计算出的调整向量不完全一致，需要进一步分析对于调整向量的取舍。

记来自 FY00 和其它组长的调整向量为 t_1，来自 FY00 和本组组员的调整向量为 t_2。基于模型假设，无人机的位置存在随机偏差，且由于无源定位无法传递距离信息，根据

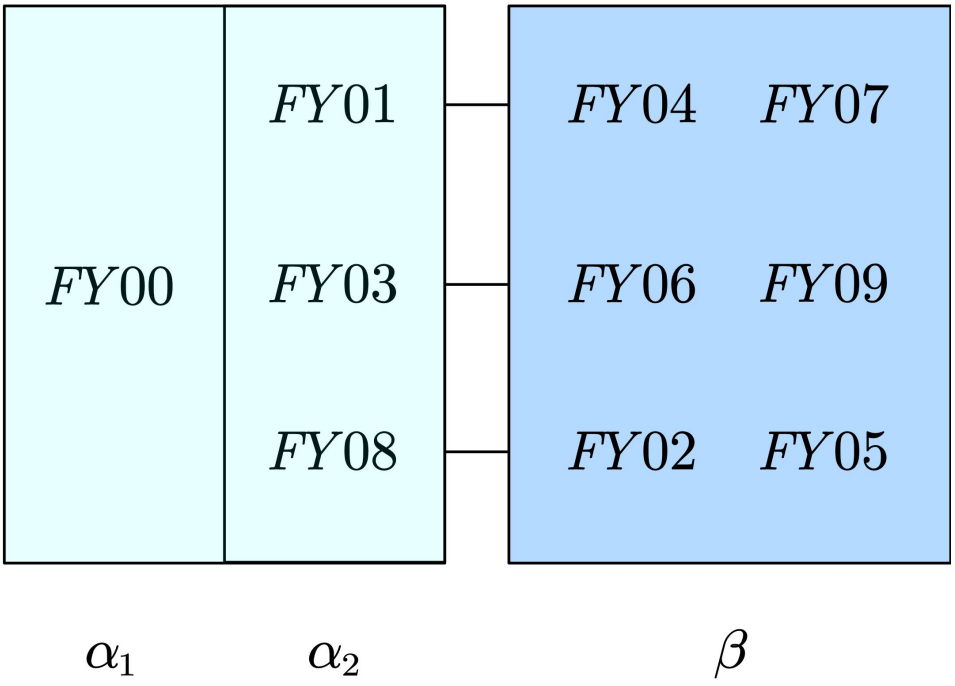


图 10　狼群层级

相似性原理与前人的研究，接收信号的无人机无法对信号源实施有效定位。因此，在组长无人机对多个调整向量进行取舍时，无法从全局出发，以 S_1^2 和 S_2^2 作为最优化指标。因此，我们考虑计算移动了一定的调整向量 t 后，与相应信号源的定位产生的径向误差 e_n 和切向误差 e_t, 并将其作为组长决定调整向量的最优化指标。为了消除正负号带来的不必要干扰，定义局部总误差为

$$E = \sum q_i(e_{ni}^2 + e_{ti}^2) \tag{11}$$

式中，q_i 是狼群的层级系数，当信号源相比于本机在狼群中的层级相同或更高时，$q_i = 1$，否则，$q_i < 1$。基于模型假设，无人机之间相距足够远，因此可以对 e_n 和 e_t 进行线性化处理，认为调整向量与真实值的误差的尺度不足以严重影响切向方向：

$$\vec{e_i} = \begin{bmatrix} e_{ni} & e_{ti} \end{bmatrix}^T = B\left(\vec{t_i} - \vec{t}\right) \tag{12}$$

式中，B 是旋转矩阵，假设本机编号为 FY0j, 则

$$B = \begin{bmatrix} \cos\frac{2j\pi}{9} & -\sin\frac{2j\pi}{9} \\ \sin\frac{2j\pi}{9} & \cos\frac{2j\pi}{9} \end{bmatrix} \tag{13}$$

为了使 E 达到最小，应有

$$\frac{\partial E}{\partial t} = 0 \tag{14}$$

联立式(11)、(12)、(13)、(14)得到最终调整向量为

$$t_0 = \frac{\sum p_i t_i}{\sum p_i} \tag{15}$$

对于组长而言，式中，若信号源为其它组长，则 $p_i = 1$；若信号源为本组组员，则 p_i 可由式(10)确定。

基于上述分析，一轮调整的过程如下：

1. 组长接收 FY00 和另外两个组长的信号，计算得出本机调整向量 t_1。
2. 组长接收 FY00 和另外两个组员的信号，计算得出本机调整向量 t_2。
3. 组员接收 FY00、本组组长、本组另外一个组员的信号，计算得出本机最终调整向量 t_0。
4. 组长根据 t_1 和 t_2，组长的本机最终调整向量 t_0。
5. 所有无人机根据计算得到的本机最终调整向量 t_0 进行移动。

6.3.3 模型求解

我们将题目描述的初始位置作为样例 1 测试分组狼群模拟退火迭代模型。图12是分组狼群模拟退火迭代模型测试样例 1 误差评估指标曲线，无人机极径方差 S_1^2 评估指标单调递减，对于无人机极径方差 S_2^2 评估指标在第三次迭代后单调递减。总体来说，在此样例中编队无人机位置偏差稳定减小，迭代 15 次左右基本实现准确调整。图11是分组狼群模拟退火迭代模型迭代 15 次时的结果，在迭代 15 次时无人机编队均匀分布在半径 $R = 112m$ 的圆上，狼群算法可以高精度实现无人机编队均匀分布在圆上，符合题意。

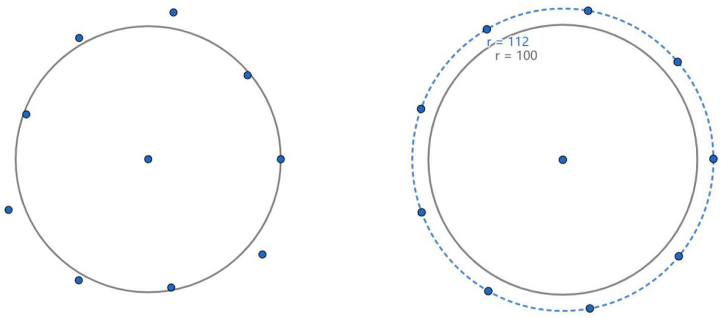


图 11　分组狼群模拟退火迭代模型测试样例 1 误差结果

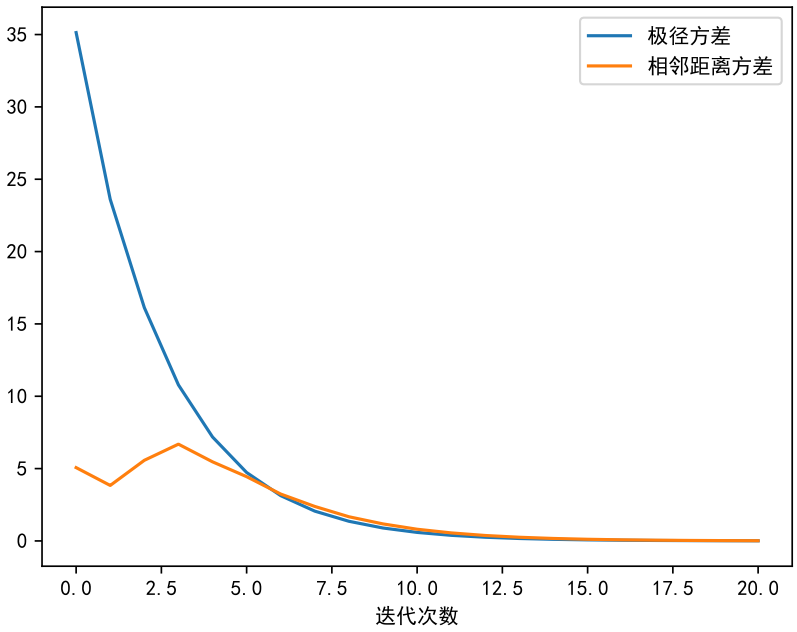


图 12　分组狼群模拟退火迭代模型测试样例 1 误差评估

考虑到真实情况中无人机的飞行会受到风速、机械结构等因素的干扰。我们在测试样例 1 的基础上增加朝向任意方向与移动距离有关的的高斯白噪声干扰作为测试样例 2。图13 是分组狼群模拟退火迭代模型测试样例 2 误差评估指标曲线。我们可以发现该曲线并没有明显的变化，迭代三次后仍然可以一致收敛，迭代 20 次左右基本实现准确调整。

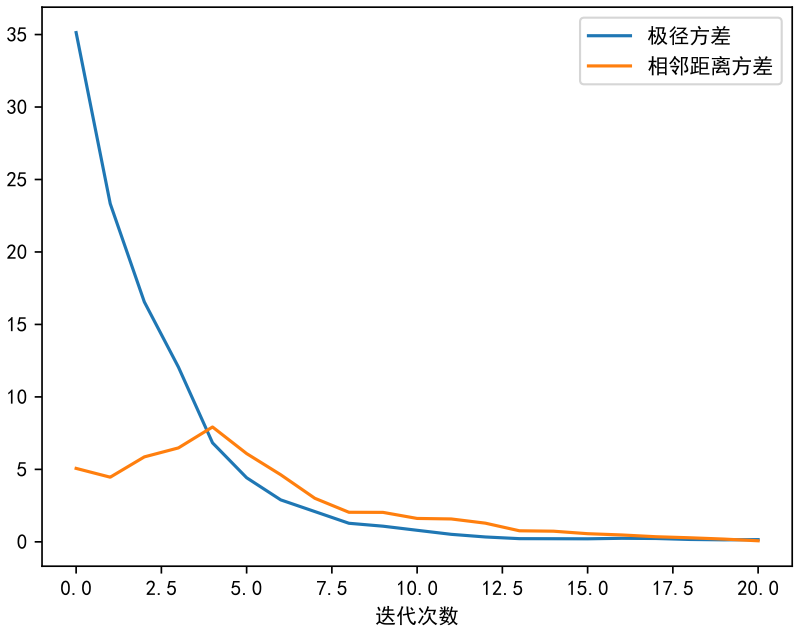


图 13　分组狼群模拟退火迭代模型测试样例 2 误差评估

6.3.4 模型评价

相对于之前两种解决方案，单源控制模型只有唯一的“头狼”，因此信息传输轮数少，但是严重受到头狼的位置误差干扰。简单迭代模型受到特定无人机的位置误差影响小，但是收敛速度慢，且收敛过程存在震荡。本文使用的改进后的狼群算法，采用“分而治之”的思想将无人机编队定位问题转化为分组定位问题，增加收敛一致性的同时减少了所需的计算量。同时，改进后头狼也可以“兼听”下属的定位信号，利用位置误差的随机性使之相互抵消。事实上，分组狼群模拟退火迭代模型在该圆形无人机编队调整中并不能保证使编队无人机分布在半径为 $100m$ 的圆上。通过我们的实践，证明了新的狼群算法结合了单源控制收敛速度快和简单迭代误差更小的优点，收敛快速且稳定。同时，该方法对随机扰动具有很好的鲁棒性。最后，本文使用分组狼群模拟退火迭代模型可以进行分组递归，通过增加层级以及层级对应的组数，来控制更大的无人机编队。

七、 问题二：模型建立与求解

7.1 模型建立：递归狼群模拟退火迭代

首先，我们假设该锥形编队分布在同一高度，则在问题一（3）中的分组狼群模拟退火迭代模型可以推广到更大的编队上。对于问题二的锥形编队，我们借鉴问题一（3）中的狼群等级设定思路：1）在编队中选取若干个圆作为基本控制单位，使用同第一问

的定位方法；2）在整个编队内，接近编队几何中心的无人机等级更高；3）在单个小组内，小组成员尽可能远离彼此；4）组长的尽量集中设置，当组长数量足够多时，将相邻组长设为广义组员，并在广义组员选取新的广义组长，以此类推。基于以上原则，最终狼群分组及等级具体设定方式如图14所示。

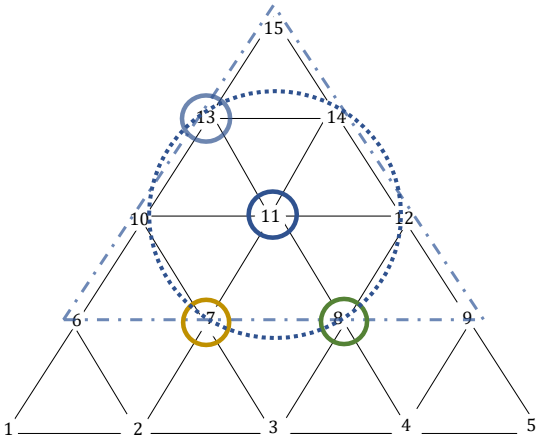


图 14　最小锥形控制

首先，在图14中，用蓝线标明了一个含有至少一个正六边形的最小锥形。在一个正六边形中，选取可能在其他六边形中心的两个无人机作为队长（若有超过两个，选择距离整体编队中心更近的一条边上的两个无人机作为组长），在此之外，选取一个可能在其他六边形中心的无人机作为副组长（如果没有，应当选取距离两个组长较远的一个无人机）。在此子图的定位过程为：组长接收副组长以及另一个组长的信号，以与第一问相同的方法计算其调整向量。组员及副组长接收两个组长的信号，以与第一问相同的方式计算其调整向量。对于最小锥形的顶点，采用中心圆圈上的副组长作为其圆心，结合圆上另外三个点向其发送的信号，计算其调整向量。

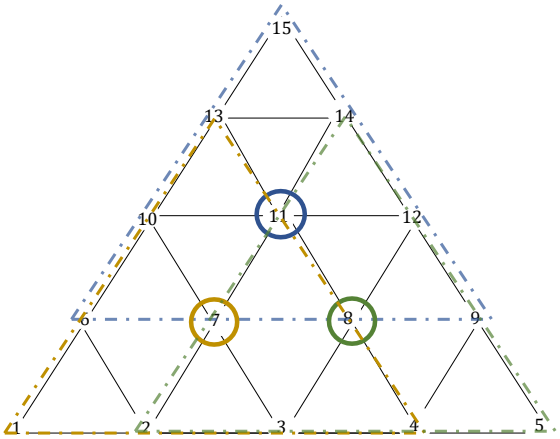


图 15　广义锥形控制

更大锥形编队的控制思路如图15所示，该图可看作由三个最小锥形组成。在三个最小锥形内，分别使用最小锥形控制的方法，计算在各个最小锥形中的调整向量。最终，使用式(15)计算其最终调整向量。可以遇见，当广义锥形的阶次再增加 2 时（当前为 5 阶），会产生一个全部由组长构成的最小锥形。此时，可以将这个最小锥形内的无人机其看作广义组员，如果想要减少因为阶次增加而带来的计算量增加，可以先在该最小锥形内选出广义组长，用相同的方式对广义组员应用一次最小锥形控制。然后，再以这些广义组员作为锥形编队外层无人机的组长，对其进行相同的最小锥形控制。从而实现对大规模锥形编队的控制。

7.2 模型求解

我们首先创建完全标准的锥形编队，然后为其中的每个无人机增加一个高斯白噪声位移，作为锥形编队的初始位置。应用递归狼群模拟退火迭代模型进行仿真，其误差评估如图16所示。可以发现，在经过了一定的震荡之后，极径方差和相邻距离方差最终收敛。我们认为，这是由于狼群模拟退火算法会使无人机均匀分布在半径一定的圆上，但由于无源定位的特点，很难确保圆的半径保持恒定。因此，在同时存在多个圆的时候，不同圆的半径变化会引起冲突，导致极径方差和相邻距离方差时大时小。同时，极径方差和相邻距离方差的收敛速度明显减慢，我们任务这是由于定位算法使用了多层递归，以及组间的协同性减少，导致收敛过程的延长。

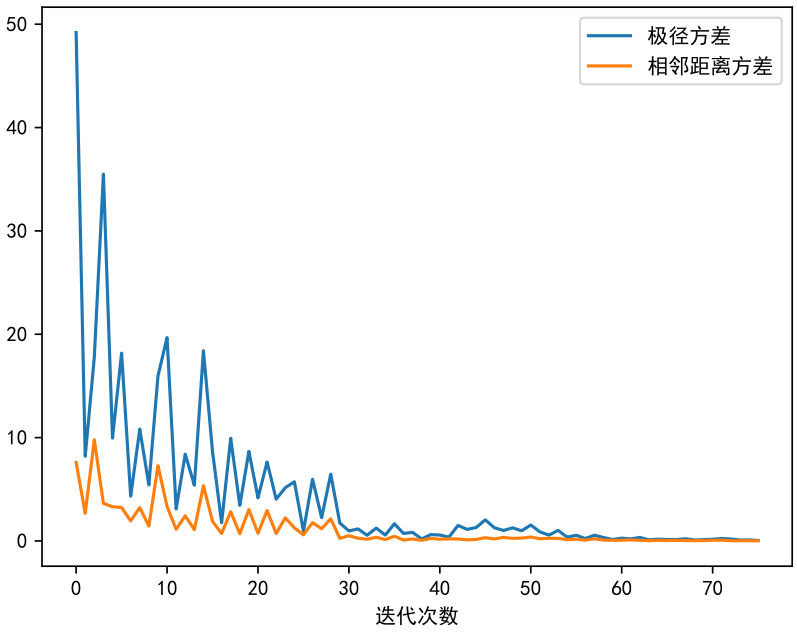


图 16

7.3 模型评价

仿真证明了分组狼群模拟退火迭代模型具有可扩展性，该方法很好的适用于锥形编队，具有误差小、收敛快速且稳定的特点。同时，面对更大规模的锥形编队，该方法可以递归的分出其锥形编队子图，运用相同的“分而治之”思想，先在子图间通过组长之间的协同，再在子图内通过组长控制其中的组员，实现大型无人机锥形编队控制。

八、 模型改进与未来工作

- 我们的程序基于 C++ 语言，接下来我们将把分组狼群模拟退火迭代模型移植到无人机单片机中，将理论结合实践进一步优化模型。
- 在建模样例测试中我们发现分组狼群模拟退火迭代模型具有适用于各种无人机编队形状的潜力，我们将进一步探索该模型适用情况。
- 在分组狼群模拟退火迭代模型测试过程中出现的测试结果，缺少严谨的数学解释，未来我们将进一步学习相关数学知识，完善数学证明过程。

参考文献

[1] 吴癸周、郭福成、张敏. 信号直接定位技术综述 [J]. 雷达学报, 2020, 9(6):16.

[2] Wei R , Beard R W . Consensus of information under dynamically changing interaction topologies[C]// American Control Conference. IEEE, 2004.

[3] Chen H , Duan H . Multiple unmanned aerial vehicle autonomous formation via wolf packs mechanism[C]// IEEE International Conference on Aircraft Utility Systems. IEEE, 2016.

[4] Bangert P . Optimization: Simulated Annealing[M]. Springer Berlin Heidelberg, 2012.

附录 A　支撑材料目录

1. 模拟无人机编队飞行代码 2. 无人机定位方程求解代码 3. 原始数据

附录 B　无人机类定义–C++ 源代码

```
// 定义单个无人机类
class Agent {
  public:
  int nam;// 无人机编号
  Point now_point;// 当前位置
  Point tar_point;// 目标位置
  Point thk_point;// 我以为我现在的位置

  Agent(int nam, Point point);
  Agent() {
    nam = -1;
  }
  Point getaNowPoint(Agent agent1, Agent agent2);// 根据两个求出一个认为自己应该去的点
  void getFinalNowPoint(Agent agent1, Agent agent2, Agent agent3);//
      根据三个分三次求出三个点，然后代数平均得到最终认为自己所在的点
  void set(int tmpnum, Point tmppoint);
  void movetoTar();// 略
  void moveVec(Vector v);// 给定向量，移动到加和
};
```

附录 C　模拟器类定义–C++ 源代码

```
// 定义模拟器类
class Stimu {
  public:
  Stimu() {
    round = 0;
    freopen("data.txt", "r", stdin);
    freopen("D:\\Project\\Python\\Modeling2022\\log.txt", "w", stdout);
    // puts("Beginning initialize Stimu");

    // 从data文件中读取初始位置
    PolarPoint tmp;
    for (int i = 0; i < 10; i++) {
      double r, theta;
      scanf("%lf %lf", &r, &theta);
      tmp.set(r, theta * DEG_TO_RAD);
```

```
        agent[i].set(i, tmp);
        if (i) {
          double r = 100;
          double theta = 40 * (i - 1);
          agent[i].tar_point.set(PolarPoint(r, theta * DEG_TO_RAD));
        }
      }
      writeLog();
      // puts("Initialization complete");
    }
    ~Stimu() {
      fclose(stdin);
      fclose(stdout);
    }
    void calAss();// 计算两个评价标准的函数
    void writeLog();// 写log日志
    /*
    log文件格式规范:
    [轮数]
    [ass1] [ass2]
    [编号] [极坐标]
    [编号] [极坐标]
    ...
    [编号] [极坐标]
    */
    void aRound();// 进行一轮模拟
    void stimulation(int num);// 模拟
    void check();
    void get_rand();// 初始化一个高斯噪声生成器

    // 模拟过程
    void grandOrder();//单源控制
    void rougLike();//简单迭代
    void wolfFuna1();// 狼群迭代


    private:
    Agent agent[10];// 定义无人机集群
    double ass1, ass2;// 两个评估标准，分别是半径方差和相邻两机距离方差
    int round;// 当前模拟的轮数
};
```

附录 D　两方向信息位置计算方法–matlab 源程序

```
function [rho, theta] = solve6(R, n1, n2, n3, a1, a2)
```

```
if a1 * a2 < 0
pd1 = n3 - n1;
pd2 = n2 - n3;

if pd1 < 0
pd1 = pd1 + 9;
end

if pd2 < 0
pd2 = pd2 + 9;
end

pd = pd1 + pd2;

if pd < 5
b = (9 - n2 + n1) * 40;

alpha1 = abs(a1) / 180 * pi;
alpha2 = abs(a2) / 180 * pi;
beta = b / 180 * pi;

f = @(alpha1, alpha2, R, beta, rho)(alpha1 + alpha2 + asin(sin(alpha2 * rho / R)) +
    asin(sin(alpha1 * rho / R)) - beta);
rho = fsolve(@(rho)(f(alpha1, alpha2, R, beta, rho)), R);
theta = 180 - abs(a1) - abs(asin(sin(alpha1 * rho / R))) / pi * 180 + 40 * (n1 - 1);

else
b = (n2 - n1) * 40;

alpha1 = abs(a1) / 180 * pi;
alpha2 = abs(a2) / 180 * pi;
beta = b / 180 * pi;

f = @(alpha1, alpha2, R, beta, rho)(alpha1 + alpha2 + asin(sin(alpha2 * rho / R)) +
    asin(sin(alpha1 * rho / R)) - beta);
rho = fsolve(@(rho)(f(alpha1, alpha2, R, beta, rho)), R);
theta = 180 + abs(a1) + abs(asin(sin(alpha1 * rho / R))) / pi * 180 + 40 * (n1 - 1);

if theta > 360
theta = theta - 360;
end

end

else
b = (n2 - n1) * 40;
```

```
if n3 < n1

alpha1 = abs(a1) / 180 * pi;
alpha2 = abs(a2) / 180 * pi;
beta = b / 180 * pi;

f = @(alpha1, alpha2, R, beta, rho)(alpha1 + alpha2 + asin(sin(alpha2 * rho / R)) +
    asin(sin(alpha1 * rho / R)) - beta);
rho = fsolve(@(rho)(f(alpha1, alpha2, R, beta, rho)), R);
theta = -abs(a1) - abs(asin(sin(alpha1 * rho / R)) / pi * 180) + 40 * (n1 - 1) + 180;

if theta > 360
theta = theta - 360;
end

else

alpha1 = abs(a1) / 180 * pi;
alpha2 = abs(a2) / 180 * pi;
beta = b / 180 * pi;

f = @(alpha1, alpha2, R, beta, rho)(alpha1 + alpha2 + asin(sin(alpha2 * rho / R)) +
    asin(sin(alpha1 * rho / R)) - beta);
rho = fsolve(@(rho)(f(alpha1, alpha2, R, beta, rho)), R);
theta = abs(a1) + abs(asin(sin(alpha1 * rho / R)) / pi * 180) + 40 * (n1 - 1) - 180;

if theta > 360
theta = theta - 360;
end

end

end

end
```

附录 E　模拟无人机编队飞行过程–C++ 源代码

```
  #pragma once
  #pragma once
  #pragma once
  #pragma once
  #include <iostream>
  #include <cstdio>
```

```cpp
#include <cstring>
#include <algorithm>
#include <cmath>
#include <random>
#include "D:\Project\Matlab\codegen\lib\solve8\solve8.h"

#define PI 3.1415926
#define RAD_TO_DEG (180 / (4 * atan(1)))
#define DEG_TO_RAD ((4 * atan(1)) / 180)
using namespace std;

// 极坐标系下点类
class PolarPoint {
  public:
  double r;
  double theta;

  PolarPoint(double tmpr = 0, double tmptheta = 0) {
    r = tmpr;
    theta = tmptheta;
  }
  void set(double tmpr, double tmptheta) {
    r = tmpr;
    theta = tmptheta;
  }
};

// 直角坐标系点类
class Point {
  public:
  double x;
  double y;
  Point(double xx = 0, double yy = 0) {
    x = xx;
    y = yy;
  }
  Point(PolarPoint p);
  void set(PolarPoint p);
  void set(Point p);
  PolarPoint toPolar();// 由直角坐标系转极坐标系
};

// 定义向量类
class Vector {
  public:
  double x;
  double y;
```

```
    Vector(double xx = 0, double yy = 0) {
        x = xx;
        y = yy;
    }
    Vector(Point a, Point b) {
        x = b.x - a.x;
        y = b.y - a.y;
    }
    double getLen();
    void show() {
        printf("%lf %lf\n", x, y);
    }
};

Vector operator * (const double k, const Vector& a)
{
    //ka
    //Pay attention to the order.
    return Vector(k * a.x, k * a.y);
}

Vector operator + (const Vector& a, const Vector& b)
{
    //a+b
    return Vector(a.x + b.x, a.y + b.y);
}
Vector operator - (const Vector& a, const Vector& b)
{
    //a-b
    return Vector(a.x - b.x, a.y - b.y);
}

bool operator == (Vector a, Vector b)
{
    return a.x == b.x && a.y == b.y;
}

double Vector::getLen() {
    return sqrt(x * x + y * y);
}

// 向量点乘
double Dot(const Vector& a, const Vector& b) {

    return a.x * b.x + a.y * b.y;
}
```

```
// 向量叉乘
double Cross(const Vector& a, const Vector& b)
{
   // |axb|
   /*
   negative when a to b is anti-clockwise;
   positive when a to b is clockwise;
   0 when a and b are on the same line.
   */
   return a.x * b.y - a.y * b.x;
}

// 第一个一定是AO，第二个是AB。AO转到AB逆时针为正
double Ang(Vector a, Vector b) {
   double ans = acos(Dot(a, b) / (a.getLen() * b.getLen()));
   if (Cross(a, b) < 0)
   return -ans;
   return ans;
}

void Point::set(Point p) {
   x = p.x;
   y = p.y;
}

// 给定两点直角坐标，返回两点距离
double getDis(Point a, Point b) {
   return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

Point::Point(PolarPoint p) {
   x = p.r * cos(p.theta);
   y = p.r * sin(p.theta);
}

void Point::set(PolarPoint p) {
   x = p.r * cos(p.theta);
   y = p.r * sin(p.theta);
}

PolarPoint Point::toPolar() {
   double r = sqrt(x * x + y * y);
   double theta = atan2(y, x);
   PolarPoint tmp(r, theta);
   return tmp;
}
```

```
// 定义单个无人机类
class Agent {
  public:
  int nam;// 无人机编号
  Point now_point;// 当前位置
  Point tar_point;// 目标位置
  Point thk_point;// 我以为我现在的位置

  Agent(int nam, Point point);
  Agent() {
    nam = -1;
  }
  Point getaNowPoint(Agent agent1, Agent agent2);// 根据两个求出一个认为自己应该去的点
  void getFinalNowPoint(Agent agent1, Agent agent2, Agent agent3);//
      根据三个分三次求出三个点，然后代数平均得到最终认为自己所在的点
  void set(int tmpnum, Point tmppoint);
  void movetoTar();// 略
  void moveVec(Vector v);// 给定向量，移动到加和
  double getRand();// 获取一个01正态分布的值

  normal_distribution<double> gaus;
  std::default_random_engine gen;// 随机因子
};

double Agent::getRand() {
  return gaus(gen);
}

void Agent::moveVec(Vector v) {
  now_point.x += v.x;
  now_point.y += v.y;
  now_point.x += v.x * 0.1 * getRand();
  now_point.y += v.y * 0.1 * getRand();
}

void Agent::movetoTar() {
  now_point = tar_point;
}

void Agent::getFinalNowPoint(Agent agent1, Agent agent2, Agent agent3) {
  Point p1 = getaNowPoint(agent1, agent2);
  Point p2 = getaNowPoint(agent1, agent3);
  Point p3 = getaNowPoint(agent2, agent3);
  Point p((p1.x + p2.x + p3.x) / 3, (p1.y + p2.y + p3.y) / 3);
  PolarPoint tmp = p3.toPolar();
  // printf("Checking::%d %lf %lf\n", nam, tmp.r, tmp.theta * RAD_TO_DEG);
  thk_point.set(p2);
```

```
    Vector mov(tar_point.x - thk_point.x, tar_point.y - thk_point.y);
    moveVec(mov);
}

// nam1 < nam2
// n2 - n1

void Agent::set(int tmpnam, Point tmppoint) {
    nam = tmpnam;
    now_point.set(tmppoint);
}

Point Agent::getaNowPoint(Agent agent1, Agent agent2) {
    double* r, * theta;
    r = new double;
    theta = new double;
    int num1 = agent1.nam;
    int num2 = agent2.nam;

    Vector v0(now_point, Point(0, 0));
    Vector v1(now_point, agent1.now_point);
    Vector v2(now_point, agent2.now_point);
    double theta1 = Ang(v0, v1);
    double theta2 = Ang(v0, v2);
    if (num1 > num2) {
        swap(num1, num2);
        swap(theta1, theta2);
    }
    // printf("Checking::in geta Now Point::agent:%d agent1:%d agent2:%d theta1:%lf
        theta2:%lf ", nam, num1, num2, theta1 * RAD_TO_DEG, theta2 * RAD_TO_DEG);
    solve8(100, num1, num2, nam, theta1 * RAD_TO_DEG, theta2 * RAD_TO_DEG ,40, r, theta);
    // printf("r:%lf theta:%lf\n", *r, *theta);
    PolarPoint tmp(*r, *theta * DEG_TO_RAD);
    Point ttmp(tmp);
    return ttmp;
}

Agent::Agent(int tmpnam, Point point) {
    nam = tmpnam;
    now_point = point;
}

// 定义模拟器类
class Stimu {
    public:
    Stimu() {
        round = 0;
```

```
    freopen("data.txt", "r", stdin);
    freopen("D:\\Project\\Python\\Modeling2022\\log.txt", "w", stdout);
    // puts("Beginning initialize Stimu");

    // 从data文件中读取初始位置
    PolarPoint tmp;
    for (int i = 0; i < 10; i++) {
        double r, theta;
        scanf("%lf %lf", &r, &theta);
        tmp.set(r, theta * DEG_TO_RAD);
        agent[i].set(i, tmp);
        if (i) {
            double r = 100;
            double theta = 40 * (i - 1);
            agent[i].tar_point.set(PolarPoint(r, theta * DEG_TO_RAD));
        }
    }
    writeLog();
    // puts("Initialization complete");
}
~Stimu() {
    fclose(stdin);
    fclose(stdout);
}
void calAss();// 计算两个评价标准的函数
void writeLog();// 写log日志
/*
log文件格式规范:
[轮数]
[ass1] [ass2]
[编号] [极坐标]
[编号] [极坐标]
...
[编号] [极坐标]
*/
void aRound();// 进行一轮模拟
void stimulation(int num);// 模拟
void check();


// 模拟过程
void grandOrder();// 冠位指定法，钦定以012画圆
void rougLike();// 肉鸽法，全随机，混沌收敛
void wolfFuna1();// 狼群


private:
```

```
    Agent agent[10];// 定义无人机集群
    double ass1, ass2;// 两个评估标准，分别是半径方差和相邻两机距离方差
    int round;// 当前模拟的轮数
};



void Stimu::wolfFuna1() {
    // 得到两个mov向量，每次按82开后加起来
    Point p1, p2;
    Vector mov1, mov2, mov;
    double k1 = 0.5, k2 = 0.5;

    p1 = agent[1].getaNowPoint(agent[4], agent[7]);
    p2 = agent[1].getaNowPoint(agent[3], agent[8]);
    mov1 = Vector(agent[1].tar_point.x - p1.x, agent[1].tar_point.y - p1.y);
    mov2 = Vector(agent[1].tar_point.x - p2.x, agent[1].tar_point.y - p2.y);
    mov = k1 * mov1 + k2 * mov2;
    agent[1].moveVec(mov);

    p1 = agent[3].getaNowPoint(agent[8], agent[1]);
    p2 = agent[3].getaNowPoint(agent[6], agent[9]);
    mov1 = Vector(agent[3].tar_point.x - p1.x, agent[3].tar_point.y - p1.y);
    mov2 = Vector(agent[3].tar_point.x - p2.x, agent[3].tar_point.y - p2.y);
    mov = k1 * mov1 + k2 * mov2;
    agent[2].moveVec(mov);

    p1 = agent[8].getaNowPoint(agent[3], agent[1]);
    p2 = agent[8].getaNowPoint(agent[2], agent[5]);
    mov1 = Vector(agent[8].tar_point.x - p1.x, agent[8].tar_point.y - p1.y);
    mov2 = Vector(agent[8].tar_point.x - p2.x, agent[8].tar_point.y - p2.y);
    mov = k1 * mov1 + k2 * mov2;
    agent[8].moveVec(mov);

    p1 = agent[4].getaNowPoint(agent[1], agent[7]);
    mov1 = Vector(agent[4].tar_point.x - p1.x, agent[4].tar_point.y - p1.y);
    agent[4].moveVec(mov1);

    p1 = agent[7].getaNowPoint(agent[1], agent[4]);
    mov1 = Vector(agent[7].tar_point.x - p1.x, agent[7].tar_point.y - p1.y);
    agent[7].moveVec(mov1);

    p1 = agent[6].getaNowPoint(agent[3], agent[9]);
    mov1 = Vector(agent[6].tar_point.x - p1.x, agent[6].tar_point.y - p1.y);
    agent[6].moveVec(mov1);

    p1 = agent[9].getaNowPoint(agent[3], agent[6]);
```

```
    mov1 = Vector(agent[9].tar_point.x - p1.x, agent[9].tar_point.y - p1.y);
    agent[9].moveVec(mov1);

    p1 = agent[2].getaNowPoint(agent[8], agent[5]);
    mov1 = Vector(agent[2].tar_point.x - p1.x, agent[2].tar_point.y - p1.y);
    agent[2].moveVec(mov1);

    p1 = agent[5].getaNowPoint(agent[8], agent[2]);
    mov1 = Vector(agent[5].tar_point.x - p1.x, agent[5].tar_point.y - p1.y);
    agent[5].moveVec(mov1);
}

void Stimu::rougLike() {
    for (int i = 1; i < 10; i++) {
        int j = (i - 1 + 4) % 9 + 1;
        int k = (i - 1 + 5) % 9 + 1;
        // int l = (i - 1 + 6) % 9 + 1;
        agent[i].thk_point = agent[i].getaNowPoint(agent[j], agent[k]);
        Vector mov(agent[i].tar_point.x - agent[i].thk_point.x, agent[i].tar_point.y -
            agent[i].thk_point.y);
        agent[i].moveVec(mov);
        // agent[i].getFinalNowPoint(agent[j], agent[k], agent[l]);
    }
}

void Stimu::grandOrder() {
    for (int i = 2; i < 10; i++) {
        if (i == 3)continue;
        agent[i].thk_point = agent[i].getaNowPoint(agent[1], agent[3]);
        Vector mov(agent[i].tar_point.x - agent[i].thk_point.x, agent[i].tar_point.y -
            agent[i].thk_point.y);
        agent[i].moveVec(mov);
    }
    return;
}

void Stimu::check() {
}
void Stimu::stimulation(int num) {
    while (num--) {
        aRound();
    }
    return;
}

void Stimu::aRound() {
    round++;
```

```
    /* 开始你的表演*/
    wolfFuna1();
    /* 表演结束 */
    writeLog();
}

void Stimu::writeLog() {
    printf("%d\n", round);
    calAss();
    printf("%lf %lf\n", ass1, ass2);
    for (int i = 0; i < 10; i++) {
        PolarPoint tmp = agent[i].now_point.toPolar();
        printf("%d %lf %lf\n", i, tmp.r, tmp.theta * RAD_TO_DEG);
    }

}

void Stimu::calAss() {
    double avg1 = 0, avg2 = 0;
    for (int i = 1; i < 10; i++) {
        avg1 += getDis(agent[0].now_point, agent[i].now_point);
        if (i > 1)
        avg2 += getDis(agent[i - 1].now_point, agent[i].now_point);
    }
    avg1 /= 9;
    avg2 /= 8;
    ass1 = 0; ass2 = 0;
    for (int i = 1; i < 10; i++) {
        ass1 += (avg1 - getDis(agent[0].now_point, agent[i].now_point)) * (avg1 -
            getDis(agent[0].now_point, agent[i].now_point));
        if (i > 1)
        ass2 += (avg2 - getDis(agent[i - 1].now_point, agent[i].now_point)) * (avg2 -
            getDis(agent[i - 1].now_point, agent[i].now_point));
    }
    ass1 /= 9;
    ass2 /= 8;
}
```

附录 F　狼群算法样例 1 迭代原始数据

```
0
35.135802 5.067676
0 0.000000 0.000000
1 100.000000 0.000000
2 98.000000 40.100000
```

```
3 112.000000 80.210000
4 105.000000 119.750000
5 98.000000 159.860000
6 112.000000 -160.140000
7 105.000000 -119.930000
8 98.000000 -79.830000
9 112.000000 -39.720000
1
18.964502 2.491173
0 0.000000 0.000000
1 104.965499 -1.938985
2 100.432327 40.801856
3 112.000000 80.210000
4 101.916435 119.056734
5 103.310326 159.789460
6 111.904057 -159.792445
7 105.098662 -121.806066
8 103.235718 -80.295622
9 111.955193 -39.796020
2
13.007097 2.517490
0 0.000000 0.000000
1 104.623986 -1.781784
2 103.668094 40.164214
3 112.000000 80.210000
4 104.939520 118.180817
5 104.030975 159.942356
6 111.967724 -159.784194
7 104.752426 -121.785543
8 104.217279 -80.022559
9 111.985622 -39.794571
3
8.675688 3.738076
0 0.000000 0.000000
1 106.174234 -2.006617
2 105.061077 40.037269
3 112.000000 80.210000
4 105.088381 118.318315
5 105.915300 159.730377
6 111.996370 -159.789458
7 106.087055 -121.960583
8 106.661267 -80.085984
9 111.997475 -39.790860
4
5.340709 3.519371
0 0.000000 0.000000
1 106.876294 -1.608948
```

```
2 107.030174 40.278732
3 112.000000 80.210000
4 106.925693 118.315565
5 107.668404 159.939035
6 111.996448 -159.788990
7 106.746662 -121.648230
8 107.550597 -80.135886
9 111.997491 -39.790752
5
3.236173 2.988516
0 0.000000 0.000000
1 108.157388 -1.488995
2 108.096899 40.120914
3 112.000000 80.210000
4 107.685006 118.675607
5 108.521299 160.004601
6 111.996492 -159.788968
7 108.097299 -121.492840
8 108.855421 -79.901852
9 111.997525 -39.790732
6
1.896608 2.078443
0 0.000000 0.000000
1 108.940342 -1.119224
2 108.952769 40.184736
3 112.000000 80.210000
4 109.009871 118.885956
5 109.395200 160.070039
6 111.996538 -159.788977
7 108.838415 -121.135981
8 109.524790 -79.905695
9 111.997557 -39.790721
7
1.198142 1.531800
0 0.000000 0.000000
1 109.597161 -0.876683
2 109.569895 40.116080
3 112.000000 80.210000
4 109.571269 119.058212
5 109.884973 159.982871
6 111.996574 -159.788989
7 109.456330 -120.916979
8 110.293902 -79.910353
9 111.997583 -39.790714
8
0.628850 1.055811
0 0.000000 0.000000
```

```
1 110.179515 -0.644977
2 110.416697 40.198174
3 112.000000 80.210000
4 110.219805 119.273799
5 110.583197 160.087042
6 111.996611 -159.789000
7 110.093480 -120.672812
8 110.623998 -79.912290
9 111.997610 -39.790707
9
0.394413 0.902821
0 0.000000 0.000000
1 110.626561 -0.541168
2 110.873769 40.222353
3 112.000000 80.210000
4 110.499227 119.445097
5 110.983395 160.152140
6 111.996648 -159.789010
7 110.396193 -120.603257
8 111.000990 -79.856109
9 111.997636 -39.790698
10
0.243327 0.614424
0 0.000000 0.000000
1 110.850240 -0.386885
2 111.110099 40.186980
3 112.000000 80.210000
4 110.905377 119.549025
5 111.203051 160.144680
6 111.996684 -159.789021
7 110.715423 -120.439351
8 111.322320 -79.833320
9 111.997661 -39.790691
11
0.111640 0.411411
0 0.000000 0.000000
1 111.220723 -0.267277
2 111.402237 40.268428
3 112.000000 80.210000
4 111.234634 119.726249
5 111.602717 160.207414
6 111.996727 -159.789033
7 111.133869 -120.291907
8 111.539069 -79.813388
9 111.997691 -39.790682
12
0.056679 0.248001
```

```
0 0.000000 0.000000
1 111.416505 -0.130568
2 111.609693 40.235388
3 112.000000 80.210000
4 111.511474 119.812588
5 111.671132 160.212186
6 111.996763 -159.789044
7 111.355530 -120.159333
8 111.703147 -79.780074
9 111.997717 -39.790675
13
0.020021 0.127282
0 0.000000 0.000000
1 111.640537 -0.028129
2 111.765484 40.248156
3 112.000000 80.210000
4 111.703552 119.930347
5 111.831992 160.217602
6 111.996800 -159.789056
7 111.630537 -120.036231
8 111.834251 -79.781517
9 111.997744 -39.790667
14
0.007854 0.077017
0 0.000000 0.000000
1 111.798387 0.039615
2 111.843535 40.252880
3 112.000000 80.210000
4 111.823583 120.013052
5 111.905806 160.228923
6 111.996836 -159.789066
7 111.748531 -119.978781
8 111.908693 -79.771694
9 111.997770 -39.790659
15
0.004488 0.045464
0 0.000000 0.000000
1 111.857367 0.086538
2 111.869302 40.224119
3 112.000000 80.210000
4 111.887595 120.051532
5 111.898389 160.219607
6 111.996865 -159.789075
7 111.803497 -119.939752
8 111.942955 -79.768783
9 111.997791 -39.790653
16
```

```
0.001584 0.019844
0 0.000000 0.000000
1 111.915966 0.123645
2 111.893082 40.224724
3 112.000000 80.210000
4 111.959784 120.101625
5 111.921002 160.215231
6 111.996901 -159.789085
7 111.906597 -119.882659
8 111.957323 -79.774033
9 111.997817 -39.790645
17
0.000274 0.008920
0 0.000000 0.000000
1 111.976790 0.152486
2 111.952206 40.223557
3 112.000000 80.210000
4 112.002200 120.145754
5 111.965049 160.220025
6 111.996942 -159.789097
7 111.976077 -119.847673
8 111.977079 -79.776349
9 111.997846 -39.790636
18
0.000175 0.004558
0 0.000000 0.000000
1 111.997174 0.178180
2 111.972496 40.226182
3 112.000000 80.210000
4 112.027181 120.165315
5 111.990700 160.219070
6 111.996977 -159.789108
7 111.993546 -119.825484
8 111.999080 -79.778596
9 111.997871 -39.790629
19
0.000085 0.002278
0 0.000000 0.000000
1 112.015689 0.189948
2 111.999597 40.223177
3 112.000000 80.210000
4 112.026050 120.185020
5 112.006200 160.220524
6 111.997013 -159.789118
7 112.012833 -119.811750
8 112.009261 -79.778721
9 111.997897 -39.790621
```

```
20
0.000149 0.001099
0 0.000000 0.000000
1 112.020023 0.201258
2 112.004876 40.220460
3 112.000000 80.210000
4 112.037676 120.195058
5 112.006239 160.220087
6 111.997049 -159.789129
7 112.012969 -119.801585
8 112.013159 -79.779939
9 111.997923 -39.790614
```