# 摘　要

随着无人机技术的飞速发展，空中交通管理成为一个关键议题。无人机飞行中的碰撞避免和航迹优化对飞行安全和效率至关重要。本文围绕五个问题，采用几何构形和人工势场方法，提出解决无人机协调避障问题的方案。

针对问题一：1 号无人机最短用时问题，在该问题中，我们构建了**协调避障几何模型**，旨在最小化 1 号无人机的飞行轨迹长度。通过几何分析，我们引出一个关键引理：在远离障碍圆圆心的位置出发的无人机，无论飞行轨迹如何，总能找到使两者连线与障碍圆相交的轨迹。这将问题转化为最小化平台 B 到平台 A 的轨迹长度，再找到适配的路径，使得 2 号无人机从平台 A 到平台 B。通过几何构形分析，我们得出结论：1 号无人机的最短航迹为平台 B 到障碍圆的切线段和一段圆弧，而另一架无人机的航迹可适配，从而 1 号无人机最短用时为 467.8 秒。

针对问题二，总用时最短问题，我们采用了**贪婪策略**和**动态规划**进行航迹规划，结合直线和圆周运动方式。为克服贪婪算法的限制，我们引入**人工势场模型**进行避障规划，将环境视为虚拟势场。通过模拟引力和斥力的作用，引导无人机规划航迹。由动态规划算法得出，两架无人机的飞行轨迹分别耗时 728 秒和 546.5 秒，其中后到达的飞行时间为 728 秒。

针对问题三：从 B 站点到圆心距离变化下的航迹问题，该问题主要考虑 B 站点到圆心距离变化时，问题一和问题二的航迹变化。通过分情况讨论，发现距离关键值为 2026 米。当距离在 (500, 2026] 范围内，两架无人机可直接按最短航线飞行；距离在 (2026,+∞) 范围内，1 号无人机需在 A 处绕飞等待，后到达目的地。

针对问题四：2 号无人机速率变化下的航迹问题，研究 2 号无人机速率在 [10,30] m/s 内变化时，问题一和问题二的航迹变化。分析发现，随着 2 号无人机速率增加，1 号无人机避让时间减少，但问题二中的总用时变化有限。当 2 号速率大于 19 m/s 时，2 号无人机最短用时为 466.3 秒，两架无人机均无需避让。

针对问题五，2 号无人机速率和 B 站点距离变化下的航迹问题，考虑 2 号无人机速率在 [10,50] m/s 内变化，且 B 站点到圆心距离在 [1,10] km 内变化。分析显示，速率较大且 B 站点距离较短时，2 号无人机到达时间较短。而速率在 10 m/s 左右，且 B 站点距离较远时，2 号无人机到达时间总体较长且随距离增加而加速增加。

综上所述，本文通过几何构形和人工势场方法解决了五个无人机协调避障问题。这些问题涵盖了无人机飞行安全和效率等方面，为未来无人机空中交通提供了重要的探讨和指导。

**关键词：最短路径 动态规划 人工势场 几何构形**

# 目录

# 1. 问题重述

## 1.1 问题背景

随着无人机技术的不断发展，其在各个领域的广泛应用愈发引人瞩目。近年来，无人机的使用量呈显著增长趋势，然而，随之而来的是愈发复杂的使用环境。尤其是在存在多样障碍物的情况下，无人机的避障航迹规划问题成为研究的热点之一。然而，实现无人机之间的协同行动却充满了复杂性，如何在保障高安全性的前提下追求高速率的飞行，成为了这一领域的难点难题。因此，探索在避开障碍物、确保无人机间不相撞的情况下，高效完成飞行任务的方法，显得尤为重要，其具有深远的研究意义和实际应用价值。

## 1.2 问题条件

平面上 A、B 两个无人机站分别位于半径为 500m 的障碍圆两边直径的延长线上，A 站距离圆心 1000m，B 站距离圆心 3500m。A、O、B 三点共线。
两架无人机（A 与 B 站的无人机分别称为“无人机 1 号”与“无人机 2 号”）分别从 A、B 两站同时出发，以恒定速率 10m/s 飞向 B 站和 A 站执行任务。飞行过程中存在以下约束条件：

- 两架无人机必须避开障碍圆。
- 两架无人机不能碰面，即两架无人机的连线必须与障碍圆相交。
- 无人机的转弯半径不小于 30m。

问题 1：本题要求 1 号到达目的站点的用时最短，并给出具体飞行方案。
问题 2：本题要求 2 号到达目的站点的用时最短，并给出具体飞行方案。
问题 3：在其他参数不变的情况下，本题需改变 B 站点到圆心的距离，分类讨论问题 1，2 中最优航迹的变化情况。
问题 4：在其他参数不变的情况下，本题需改变 B 机飞行的恒定速率（在[10,30] m/s 内变化），分类讨论问题 1，2 中最优航迹的变化情况。
问题 5：在其他参数不变的情况下，本题需改变 B 机的恒定速率（在[10,30] m/s 内变化）与 B 站点到圆心的距离（[1,10] km 内变化），讨论问题 2 中最优航线的变化情况。

# 2、问题分析

## 2.1 问题一分析

问题一旨在通过航迹规划使得 1 号无人机尽快到达目的地。通过几何模型分析，考虑 A、B 两个无人机站点与障碍圆的相对位置。具体而言，A 站距离圆心 1 千米，B 站距离圆心 3.5 千米。通过几何推导，我们得出了重要结论：为最小化 1 号无人机到达时间，需要选择离障碍物较远的无人机站点，而忽略较近的站点。基于此，我们提出了一条航迹

规划：从圆外某点出发，与圆相切，穿过两站点，再次与圆相切并达到目的地。最终，我们运用几何计算求解了该路径上的时间。

## 2.2 问题二分析

问题二要求在 1 号无人机到达目的地的情况下，规划 2 号无人机的最短时间航迹。我们采用了贪婪策略与动态规划相结合的方法。首先，构建基本的无人机移动模型，考虑速度、角速度及站点距离等因素，作为动态规划的起始路径。引入人工势场法，通过引力和斥力场，以及目标引力场、障碍圆斥力场和相对位置调节场，将问题转化为势场下的路径规划问题。这样，无人机可以按照势场进行航迹规划，同时满足约束条件。通过求解模型，我们得到了无人机的最短路径和时间。

## 2.3 问题三分析

问题三考察了 B 站点与圆心距离对问题一和问题二航迹规划的影响。对于问题一，我们通过改变 B 点距离圆心的步长，分析不同距离下的几何特性变化，同时测试最优航迹。对于问题二，同样利用人工势场模型，代入不同距离值，对结果进行比较分析。

## 2.4 问题四分析

问题四要求在 2 号无人机的恒定速率在[10,30] m/s 内变化的情况下，探究问题一中航迹的变化。在问题一中，1 号无人机绕圆避让，2 号无人机直接前往目的地。增加 2 号无人机的速率只会减少 1 号无人机的避让时间。同时，我们需要考虑 2 号无人机速率变化对问题二最优航迹的影响。存在一个临界速率，超过此值后，2 号无人机可在 1 号无人机到达点 E 前到达安全位置，避免碰撞。通过建立问题二模型，求解这个临界速率。

## 2.5 问题五分析

问题五考虑了 2 号无人机的恒定速率在[10,50] m/s 内变化以及 B 站点到圆心的距离在[1,10] km 内变化的情况，探究问题二最优航迹的变化。我们分析了两因素的综合影响：2 号无人机速率与 B 站点距离。对问题二的模型，区间划分不同速率和距离组合，探讨其对航迹规划的影响。这两因素决定了追赶点位置，影响 1 号无人机是否绕圈避让，2 号无人机是否避让。不同情况下，两无人机分为三种飞行方式。每速率对应特定距离，实现 2 号无人机最短到达时间。

# 3、 模型假设

为了构建更为准确的数学模型，本文结合实际生活情况，在排除一些不可控因素的干扰后，本文做出以下假设或约束：

1. 本文不考虑无人机本身体积质量对路径规划的影响，假设无人机可以无限贴近障碍圆。
2. 无人机以及障碍圆都在同一个二维欧式平面中。

# 4、 符号说明

| 符号 | 说明 |
|---|---|
| $r$ | 无人机转弯半径最小 |
| $r_{min}$ | 转弯半径线速度 |
| $U_{att(z)}$ | 目标点位置 |
| $H_1$ | 1 号无人机运动点 |
| $V_1$ | 2 号无人机运动点 |
| $V_2$ | 2 号无人机运动点在 BE 上的从动点 |
| $\rho(\theta)$ | PQ 的极坐标方程 |
| $S_1$ | 过$V_1$做圆 O 的切线$S_1$ |
| $S_2$ | 过$H_1$做圆 O 的切线$S_2$ |
| $T(x)$ | 1 号无人机到达目的站点的最短时间函数 |
| $V_2$ | 2 号无人机运动点在 BE 上的从动点 |

# 5、模型建立与求解

## 5.1 问题一模型的建立

平面上有一个半径为 R 的障碍物圆 O，其直径延长线的两端分别有两个平台 A 和 B。无人机 P 从平台 A 飞向平台 B，无人机 Q 从平台 B 飞向平台 A，两架无人机的速度恒定。场景如图 1 所示。

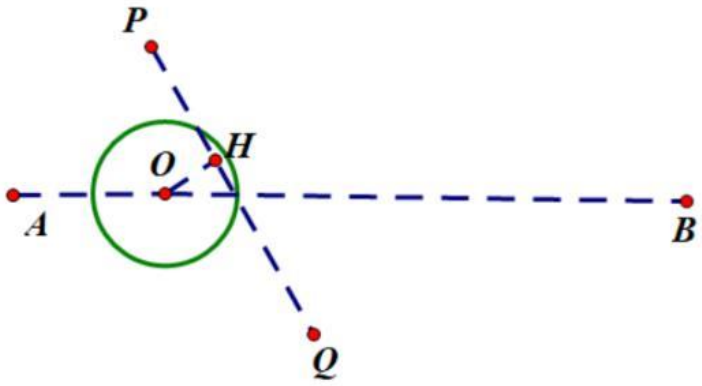



图 1 无人机平台和障碍物圈示意图

本文令 A 平台到障碍圆圆心的距离（即 OA）长度为$l_a$，B 平台到障碍圆圆心的距离（即 OB）长度为 $l_b$，障碍圆的半径为 $R$，无人机的恒定速度为 $V$，无人机 P、Q 的飞行轨迹路径长度为 $l_1$和 $l_2$。

该问要求 1 号到达终点的无人机飞行所花费时间最短，由于无人机飞行速度恒定，所以飞行时间与飞行轨迹长度具有线性关系，问题也等价于求解 1 号到终点无人机的最小飞行轨迹长度，即该问的优化目标函数可以表示为

$$min\ \{min\, L_i\}$$

同时，模型应当满足以下限制条件：

1.两无人机需要避开障碍圆，即：

$$OP > R\ , OQ > R$$

2.两无人机不能碰面，故两架无人机的连线与圆相交，即：

$$PQ \cap \odot O \neq \emptyset$$

该条件等价于障碍圆圆心到无人机连线的距离 OH 小于障碍圆半径 R，并且无人机位于垂足 H 的两侧。

3.无人机转弯半径大于其下限，即

$$r \geq r_{min}$$

根据角速度公式，我们可以得到最大角速度

$$\omega_{max} = \frac{v}{r_{min}}$$

其中， $v$为线速度， $r_{min}$是最小转弯半径（也就是最小曲率圆半径）。两架无人机只能分别位于 AB 连线的上下方，因为如果无人机位于 AB 同一侧，两架无人机在相向飞行的过程中，为了不碰面，一定会有至少一个时间点，一架无人机在另一架无人机的正上方，也就是两架无人机连线延长线垂直于 AB，此时，无人机的连线必然不与障碍圆相交，与题目限制条件矛盾。示意图如下

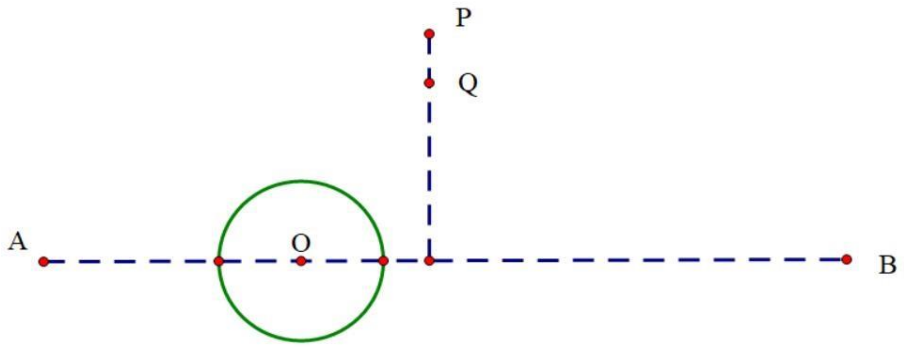



图 2 无人机同侧飞行示意图

鉴于在前面所提及的约束条件中，仅条件 2 具有一定复杂性，因其涉及两架无人机的协同系统，我们首先对该条件进行了转化和处理。

我们假设 A 站点距离圆心 O 比 B 站点更近。在两架无人机速度相同的情况下，从 A 站点出发的无人机可以凭借其较小的距离优势，更快地调整以 O 为中心的极坐标角度，同时以相同的程度控制其与圆心的距离。与此同时，从 B 站点出发的无人机在离障碍圆较远时，A 站点的无人机可以通过暂时的悬停或轻微调整，等待其靠近，并开始绕 O 驶向终点，从而保持两架无人机的连线始终经过障碍圆。基于上述推理，我们得出以下引理：出发自离障碍圆圆心较远站点的无人机，无论其飞行轨迹如何，始终可以通过适当的轨迹调整，使其与出发自另一站点的无人机保持连线交于障碍圆。

若从 A 到 B 的路径距离最短，根据上述引理，无人机同样可以沿此最短路径从 B 到 A，然而若从 B 到 A 的路径距离最短，则不一定适用，因为当 $a$ 相对于非常大的值时，无人机可能会过早飞越障碍圆，使得两架无人机连线无法再与圆相交。因此，在寻求最优路径时，只能选择从 B 站点出发的无人机作为 1 号到达目标地点。

综上所述，此问题可以被转化为先求解从 B 到 A 的路径，该路径满足 5.1.1 中，然后再找到一个适配的从 A 到 B 的路径的单无人机系统最短航迹规划

为求解单无人机系统最短航迹，本文考虑满足前文中约束 1 的最短路，给出如图所示临界情况。

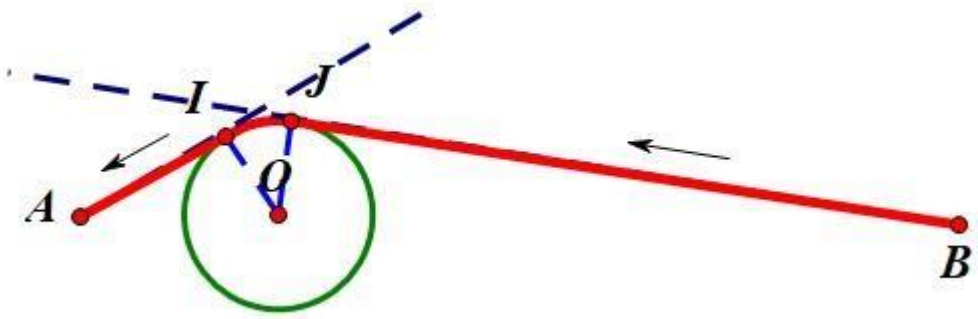



图 3 临界路径示意图

从点 A 和 B 分别与圆心 O 作切线，使得无人机在切线上出发，然后在 B 站点与圆弧切点 J 附近沿着障碍物圆弧飞行，在达到 A 站点与圆弧切点 I 后，再沿着切线前往 A 站

点。这种方案同时满足了 5.1.1 节中的约束条件 3。接下来，我们将证明这种方案是最优的，即不存在比这种方案更短的航迹长度。

我们可以用折线来初步表示无人机的飞行轨迹，而在转折点处使用半径大于或等于的圆，或者曲率小于的光滑曲线。讨论切线段与弧长之间的关系，由文献[2]我们可以得知，从圆外或者光滑曲线外的一点引出两条切线段，切线所夹的弧长小于两条切线段之和。

在切点和切线段不变的情况下，光滑曲线的曲率越小（即曲率圆的半径越大），曲线的长度越小（在极限情况下，曲线变成直线，形成一个三角形）。因此，可以得出结论，令无人机沿着圆弧飞行是合理的，而且这种方案的长度是最短的。本文将分三种情况讨论以证明这种路径的合理性。只有一条切线的单折线情况

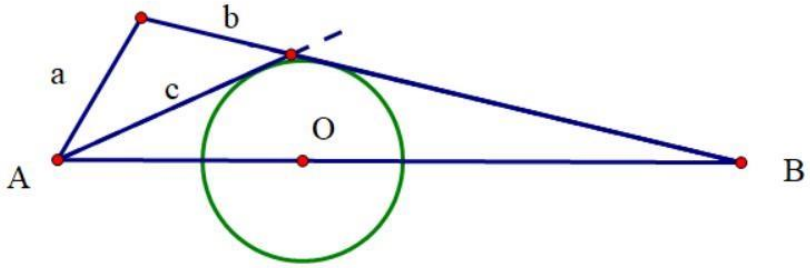



图 4 情况 1 示意图

无人机沿着 a 和 b 折线行驶，中间在沿着一个圆弧转弯。三角形两边之和大于第三边，即使再沿着圆弧行驶，甚至将 a 和 b 转化为光滑曲线，路径长度也会大于 c。

1.无切线的单折线情况

根据图示，过一条切线作延长线，将该情况转化为情况 1 进行讨论，显然该路径不如转化后的情况 1 短，所以该情况不被考虑。

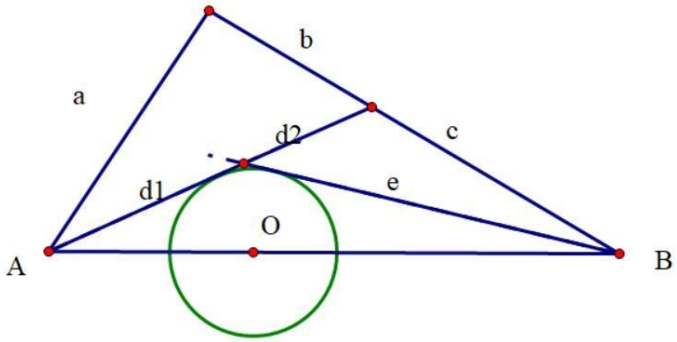



图 5 情况 2 示意图

2.多折线情况

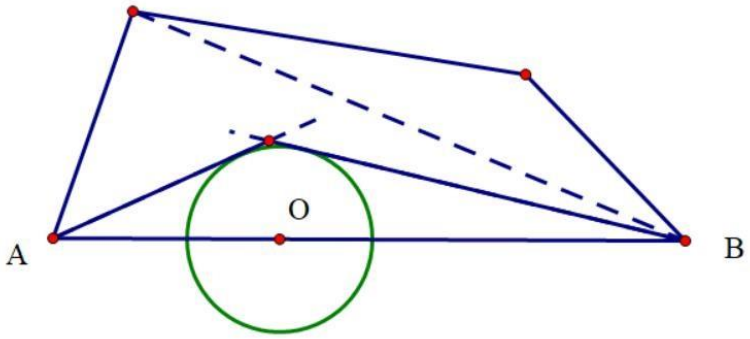



图 6 情况 3 示意图

结合情况 2，易得该种情况下的折线长度是小于本文所提出解的长度的。随着折线数不断增加，会无线接近于曲线，所以曲线代替符合问题限制条件的已知可行点间直线并不是一个可行的方向。

结合三种情况和切线与所夹弧的关系，增加折线减少相切点线并不能使得结果更优，都可能可以说明本文所提出无人机运行的方案是最优的。

## 5.2 首架无人机最短用时问题求解

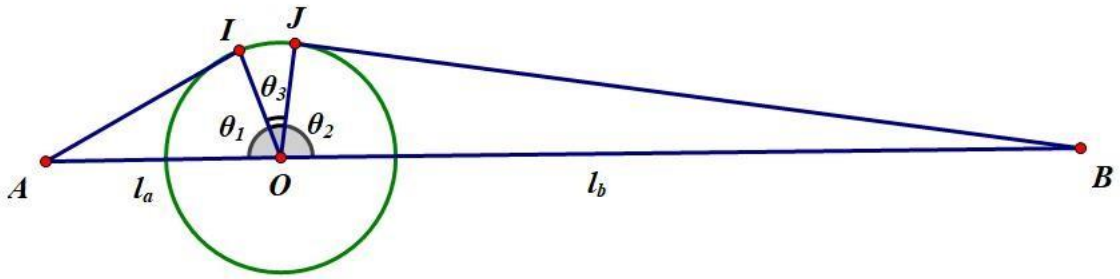



图 7 飞行轨迹与圆内夹角示意图

如图所描述，在无人机的航迹中，我们定义切点与圆心的连线为 OI 和 OJ，同时将站点之间的连线记作 AB。对于这些情况，我们分别用 α 表示切点与圆心连线 OI 与站点连线 AB 所夹的角度，用 β 表示切点与圆心连线 OJ 与站点连线 AB 所夹的角度，而 γ 则表示 OI 与 OJ 之间的夹角：

$$\alpha = \arctan\left(\frac{r}{d}\right),$$

$$\beta = \arctan\left(\frac{r}{d'}\right),$$

$$\gamma = |\alpha - \beta|.$$

接下来，我们可以运用这三个角度来计算 1 号无人机从起始点到目的地的最短路径长度。

$$L = l_a sin\theta_1 + l_b sin\theta_2 + R\theta_3$$

为了得到两架无人机的可行飞行轨迹，只需再找到一条满足条件的从点 A 到点 B 的无人机路径。示意图如下所示：

为获得两家无人机的可行飞行轨迹，只需再寻找一条符合条件的从 A 到 B 的无人机路线即可，示意图如下

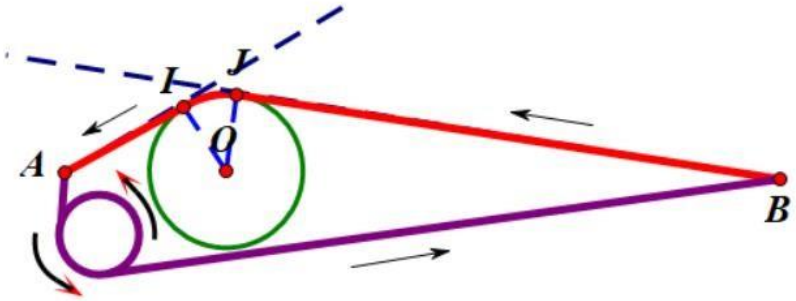



图 8 问题一路线求解示意图

图示给出了一条满足要求的无人机航迹，其中 1 号无人机从站点 A 出发，在 A 下方的区域绕一个较小半径的圆盘旋转两周，然后直线飞向站点 B。在整个过程中，两架无人机的连线始终与障碍圆相交。

在具体数值上，我们取障碍圆的半径 $R = 500$ 米，OA 的长度为 1 千米，OB 的长度为 3.5 千米，无人机的恒定速率为 10 米/秒，最小转弯半径为 30 米。通过计算，我们得出路径 $AI$ 的长度为 864.671 米，路径 JB 的长度为 3462.184 米，路径 $IJ$ 之间的弧长为 335.368 米，从而无人机的总飞行距离为 4636.4 米。1 号无人机按照该路径飞行所需的时间为 467.83 秒。

# 6. 问题二模型的建立与求解

对于问题二，本文的目标是规划出一条用时最短的路径，为了达到这个目标，我们首先建立了一个基本的无人机移动模型作为动态规划的基础。然后，我们采用了人工势场法，将目标引力场、障碍圆斥力场以及相对位置调节场这三种势场引入进来，确定在每个时间点的运动方向。通过这些势场，我们把问题转化为了不同势场之间的相互作用分析。

## 6.1 无人机航迹规划的移动模型

针对问题二，本文的目标是寻找用时最短的路径规划。为了实现这一目标，首先构建了一个基本的无人机移动模型，以作为动态规划的基础。接着，采用人工势场法引入了三种势场：目标引力场、障碍圆斥力场和相对位置调节场，用以确定每个时间点的运动方向。这些势场的引入使问题转化为了不同势场之间的相互作用分析。

在解决这一问题时，为更准确地描述和规划无人机的动态飞行路径，我们将时间离散化，构建了无人机移动模型。考虑到无人机的运动特性，我们将固定速度的无人机运动轨迹分为两类：直线匀速运动轨迹模型和恒定圆弧运动轨迹模型。

设每个无人机的二维状态点为 $P$，直线和圆弧的恒定速度分别为 $v_{\text{line}}$ 和 $v_{\text{arc}}$，无人机动力学离散时间步长为 $\Delta t$，则直线匀速运动轨迹模型为：

$$P_{\text{n+1}} = P_{\text{n}} + v_{\text{line}} \cdot \Delta t \cdot \cos(\text{x}_n),$$

$$x_{(n+1)} = ax_n,$$

而恒定圆弧运动轨迹模型为：

$$\sin\left(\mathrm{x}_{(n+1)} + \omega \cdot \triangle t\right) - \sin(\mathrm{x}_n)$$
$$-\cos(\mathrm{x}_n + \omega \cdot \triangle t) + \cos(\mathrm{x}_n)$$
$$\mathrm{x}_{(n+1)} = \mathrm{x}_n + \omega \cdot \triangle t,$$

其中，$\omega$ 是角速度,圆弧半径$r \geq r_{min}$。

综合上述公式，我们得到针对无人机的移动模型为：

$$\mathrm{x}_{(n+1)} = \mathrm{x}_n + \omega \cdot \triangle t,$$

其中，$v$ 和 $\omega$ 是根据具体情况选取的速度和角速度。对于无人机这一动态目标，其状态点 $P$ 位于二维状态空间中，遵循无人机的移动模型。

这一移动模型的引入，有助于更准确地描述无人机的运动，为后续的路径规划提供基础。

## 6.2 人工势场模型

为了在满足约束条件的前提下尽快使两架无人机达到目标点，本文采用人工势场模型进行避障，其基本原理是将无人机所处环境视为虚拟势场，并通过模拟引力和斥力等相互作用来引导无人机的航迹规划。无人机的运动被视为在特定势场函数中进行，该势场由三个关键部分组成：目标引力场、障碍圆斥力场和相对位置调节场。接下来对这三种势场进行建模：

### 6.2.1. 目标引力场

连接无人机与目标点的关系通过引力势场函数来体现，该引力场会吸引无人机朝目标点前进。在此，我们选择了一个距离线性增长的引力势场，即所谓的圆锥形势场。无人机与目标点之间产生的吸引力势场函数表示为：

$$F_{att} = \eta \cdot \left(P_{goal} - P_{drone}\right)$$

其中，$\eta$ 是引力场增益系数，$P_{goal}$ 表示目标点，$P_{drone}$ 表示无人机位置。在引力场内，无人机受到的吸引力由梯度负方向的引力势场函数给出：

$$F_{att}(x) = -\nabla U_{attr}$$

### 6.2.2. 障碍圆斥力场

无人机与障碍圆的关系通过斥力势场函数来表现，斥力使得无人机在接近障碍物时受到排斥，从而避免碰撞。斥力场需满足以下条件：当无人机靠近障碍物时，斥力应足够强以防止碰撞；随着无人机与障碍物距离增加，斥力逐渐减弱至零，使得无人机在远离障碍物区域内可以自由移动。

定义 $r_{\{max\}}$ 阻碍圆的最大影响距离，当无人机与障碍圆的圆心距离大于 $r_{\{max\}}$ 斥力不影响无人机。因此，无人机与障碍圆之间的斥力势场函数表示为：

$$U_{rep}(x)=\begin{cases}\frac{1}{2}\beta\left(\frac{1}{\|x,z\|}-\frac{1}{d_0}\right)^2 & \|x,z\|\le d_0\\ 0 & \|x,z\|>d_0\end{cases}$$

其中，$\beta$ 是斥力增益系数，$Z$ 表示障碍圆的位置。

$$F_{rep}(x)=-\nabla U_{rep(I)}$$

需要注意的是，斥力势场采用了与距离平方反比的势能形式，而非简单的反比形式，以保证分段的作用力函数连续，避免作用力突然跃变，从而避免出现不连续的航迹。

综上所述，本文通过构建目标引力场和障碍圆斥力场，以及考虑相对位置调节场，实现了两架无人机在人工势场模型下的航迹规划，以尽快到达目标点并避开障碍物。障碍圆斥力场

无人机与障碍圆之间的关系被表现为斥力势场函数，斥力函数使得无人机在靠近障碍物时感受到斥力，从而避免与障碍物碰撞。

斥力场应当满足如下限制：当无人机靠近障碍物时，斥力应足够强大，使无人机能够避免与障碍物碰撞；并且，随着无人机与障碍物的距离增大，斥力应逐渐减弱至 0，以便无人机在远离障碍物的区域内能够自由移动。

### 6.2.3.相对位置调节场

根据题目要求，两架无人机之间的连线必须与障碍圆相交，这意味着连线的圆心到直线的距离必须小于障碍圆的半径。为了实现这一目标，本文试图通过调节势场函数来实时调整两架无人机之间连线与障碍圆的关系，以保持稳定的相交状态。

基于前述分析，我们构建了一个相对位置调节场，其特点如下图所示：

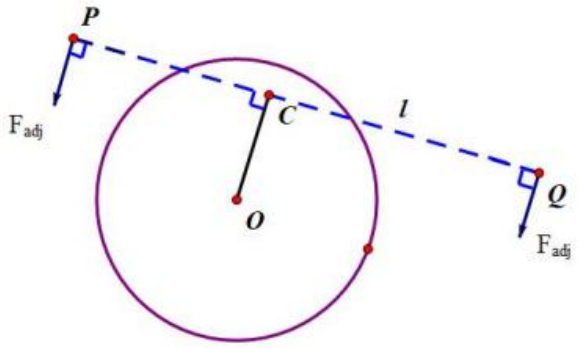



图 9 调节势场作用示意图

调节势场函数在整个平面上分布，而调节势场的等势线与无人机之间的连线平行，从而确保无人机之间的连线始终在同一势能线上。在障碍圆的边缘处，调节场的电势应趋近于无穷大，以确保无人机之间的连线始终与障碍圆相交。而在障碍圆心的半径为 $\frac{R}{2}$ 的区域内，调节势场函数为零，保证连线不受影响。

$$U_{rep}(x)=\begin{cases}\frac{1}{2}B\left(\frac{1}{R-D}-\frac{2}{R}\right)^2 & D(O,I)\in\left[\frac{1}{2}R,R\right]\\ 0 & D(O,I)\in\left[0,\frac{1}{2}R\right]\end{cases}$$

其中，$B$为相对位置调节增益系数，$D$为点到直线的距离，$D(O,I)$为障碍圆圆心到直线 L 的距离。

$$F_{adj}(x)=-\nabla U_{adj(I)}$$

这个调节力在距离障碍圆边界最大距离的范围内变化迅速，有效地将无人机之间的连线控制在一定的范围内。

综上所述，三种势场相互影响，通过合力将无人机引导至特定的路径。无人机在整个环境中移动时，受到这三种势场的综合影响，其合势场函数和合力可以表示为:

$$\begin{cases}U_t(x)=U_{att}(x)+U_{rep}(x)+U_{adj}(x)\\ F_t(x)=F_{att}(x)+F_{rep}(x)+F_{adj}(x)\end{cases}$$

通过这三种势场的综合作用，我们能够设计出无人机在环境中移动的路径。

## 6.3 无人机最短总用时问题求解

本问题的目标是在满足约束条件的前提下，求解两架无人机中后到达的飞行轨迹，使总飞行时间最小化。这里两架无人机的速度相同，因此目标函数可表示为：

$$\text{Min } T=\max\{T_1,T_2\},$$

$$T_i=\frac{d_i}{V},$$

其中 $T_i$ 表示第 $i$ 架无人机的飞行时间， $d_i$表示第 $i$ 架无人机的飞行距离， $V$ 表示无人机的速度。

本问题受到如下约束条件：

1. 两架无人机连线与障碍圆相交，即障碍圆圆心到连线的距离小于障碍圆半径。

2. 无人机与障碍物不发生碰撞。

3. 无人机的最小转弯半径不小于 30 m。

为了满足约束 1，本文引入了相对位置调节场，以限制无人机连线与障碍圆之间的距离。此外，为满足约束 3，本文构建的人工势场受力连续，确保航迹的平滑性，从而满足最小转弯半径的要求。

在人工势场法中，涉及四个势场参数：引力场增益系数，斥力场增益系数，相对位置调节增益系数，障碍圆最大影响距离。这些参数需要根据实际情况进行调节。

具体算法流程如下：

Step 1. 初始化势场参数。

Step 2. 初始化两架无人机位置。

Step 3. 根据引力、斥力和相对位置调节场计算合力大小和方向。

Step 4. 根据基础无人机航迹规划移动模型计算下一个航迹点。

Step 5. 判断航迹点是否满足不碰撞障碍物的约束。若满足，则输出方案；若不满足，则调整斥力场增益系数，返回步骤 2。

Step 6. 保存当前无人机位置，设为起点，判断是否到达目标点。若未到达，返回步骤 3；若到达，结束航迹规划。

通过这一算法，可以获得两架无人机的最优飞行路径，以满足约束条件并使总飞行时间最小化。

我们对相对位置调节增益系数从 0.01 到 0.2 进行搜索，规划其对应的航迹，并计算对应的离障碍圆圆心的最近距离和总时间得到如图 11 和图 12 所示结果。值得一提的是，本文将两架无人机初始位置分别向上和向下调整了 1m，确保两者不会陷入“死锁”的局面。

表 1 问题二参数选择

| 参数 | 引力场增益系数$a$ | 相对位置调节增益系数 B | 障碍圆最大影响距离$d_0$ | 采样时间间隔$\Delta(t_s)$ |
|---|---|---|---|---|
| 取值 | 1 | 0.05 | 2 | 0.1 |

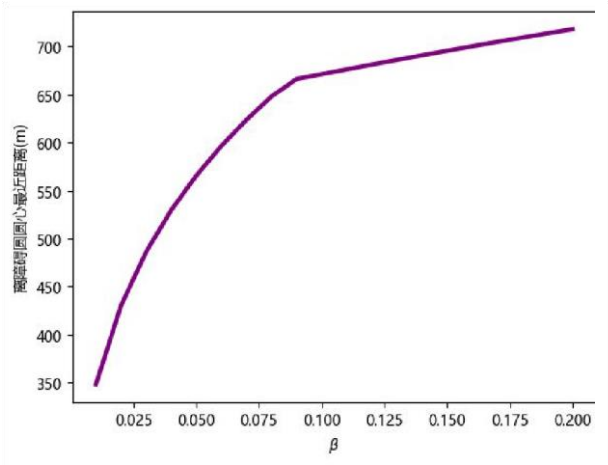



图 10 斥力系数与离圆最近距离关系图

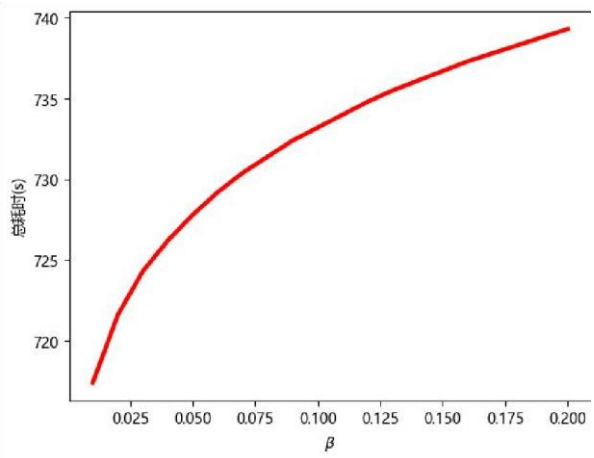



图 11 斥力系数对应飞行总耗时

从图中我们可以观察到，随着斥力的增强作用，无人机离障碍圆圆心的最近距离逐渐增加。与此同时，两架无人机的总飞行时间也随之增加，这与我们的预期相符。当斥力场增益系数 $k_{\mathrm{rep}}$取值为 0.04 时，无人机离障碍圆圆心的最近距离超过了 500 米，随后的轨迹均满足约束条件。因此，在保持约束条件的前提下，选择 $k_{\mathrm{rep}}=0.04$ 时，可以实现总飞行时间的最小化。在这种情况下，两架无人机的飞行路径如下图所示，总飞行时间为 728.8 秒。

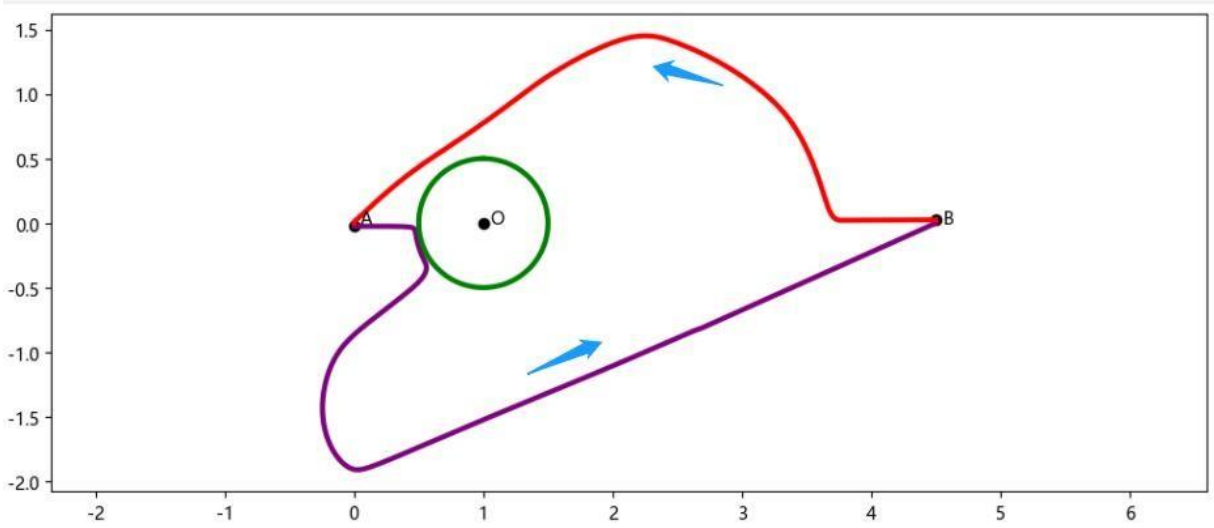



图 12 斥力场无人机飞行轨迹图

在图中，从站点 B 出发的无人机首先直线飞向站点 A，然后经过一段较长的绕行路径抵达 A，总用时为 546.5 秒。与之相对，从站点 A 出发的无人机则先直线飞向站点 B，然后经过一段较长的绕行路径回到 B，总用时为 728.8 秒。因此，总的飞行时间（即两架无人机中后到达的飞行时间）为 728.8 秒。

# 7. 问题三求解

本问需要在其它参数不改变的前提下，分析站点 B 到圆心 O 距离变化对问题一和二无人机最优航迹的影响。在对问题一影响的分析中，我们对相对较长站点变化做出讨论；对问题二影响的分析中，我们比较不同的 B 站点与圆心距离对于人工势场求解和结果变化的影响。

## 7.1 对问题一的影响

根据问题一中提出的引理，即离障碍圆圆心较远站点出发的无人机无论飞行轨迹如何，在另一站点起飞的无人机总有合适的飞行轨迹使两架无人机连线始终与障碍圆相交。

再结合后续对近端最远距离的讨论，可以知道距离圆心较远端站点起飞的无人机先到达目的地。

在本问，B 距圆心距离是一个可变参数，所以我们需要对 B 站点和 A 站点与圆心距离的相对大小进行讨论。

当$l_b > 1km$时，依旧是 B 站点起飞的无人机先到达目标点。两架无人机可以按照问题一所给轨迹飞行。最优轨迹距离与 B 站点到圆心距离关系图如下：

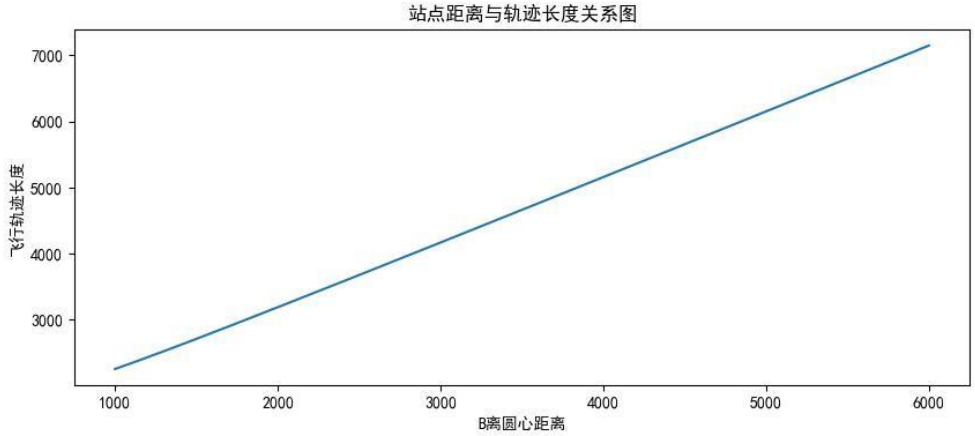



图 13 较长站点距离与轨迹长度关系图

根据观察可以得知，在 B 站点为较远端的前提下，当 B 聚增大时，最优飞行轨迹长度几乎是线性增加的。

当 $l_b = l_a = 1km$ 时，两家无人机可以上下对称形式按照最优轨迹飞行至终点，示意图如下：

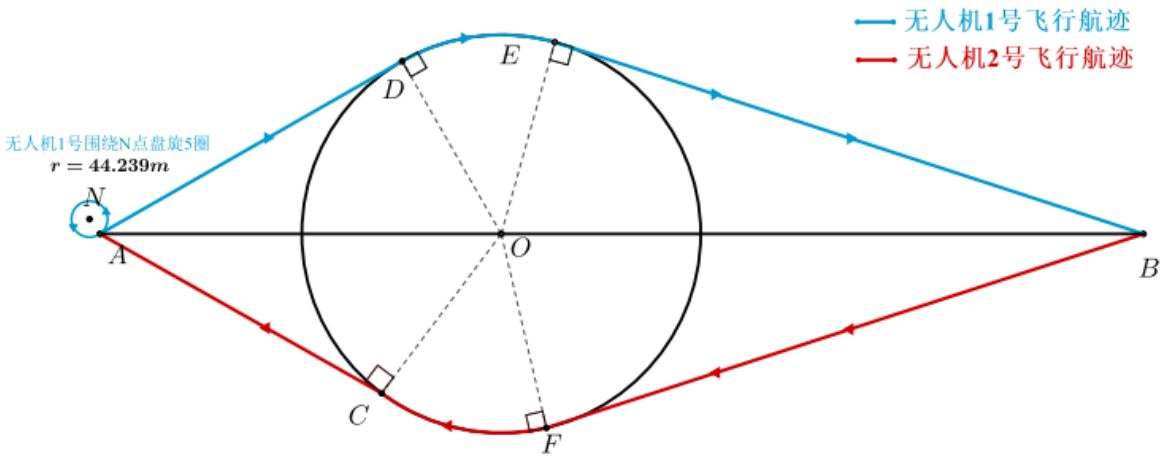



图 14 两站点行距相等最优轨迹图

该情况下两架无人机连线始终过圆心，能够满足三个条件，并且能同时到达目标点，是一种最优临界状态。可以求得该情况下的最优轨迹长度为 2255.65m，先到达终点无人机的飞行时长为 225.56s。

当 $l_b < l_a = 1km$ ，A 站点到圆心距离相对较远，从 A 点出发的无人机按最优路径先到达目标点。其最优轨迹距离与 B 站点到圆心距离关系图如下：

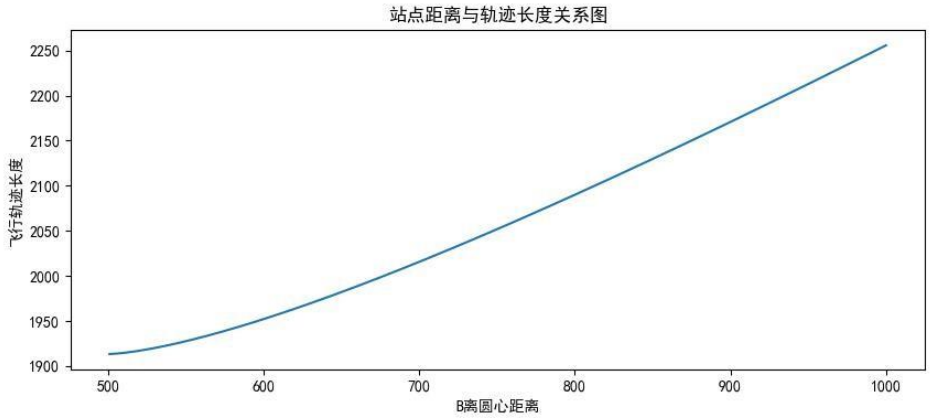



图 15 较短站点距离与轨迹长度关系图

通过观察可以得知，当 B 和圆心的距离的距离在 500m 的临界状态到 1000m 之间，问题一最优轨迹长度总体还是表现为近似的线性关系。

## 7.2 对问题二的影响

在问题二中，我们通过动态规划得出了在给定 OB 长度为 3.5 km 情况下的最优航迹。从站点 B 出发的无人机首先以直线飞行方式前往站点 A 一段距离，然后选择绕行

较大的弧线路径，最终抵达 A。同时，从站点 A 出发的无人机则开始以直线飞行方式前往站点 B 一段距离，然后逐渐转向一个向后的弧线路径，最终直线飞向 B。

在本问题中，站点 B 距圆心距离是一个可变参数，因此我们仍需对站点 B 和站点 A 与圆心距离的相对大小进行讨论。本文尝试了 OB 值为 1.5 km、3 km 和 4.5 km 的情况，分别对应不同的最优航迹，如下图所示。

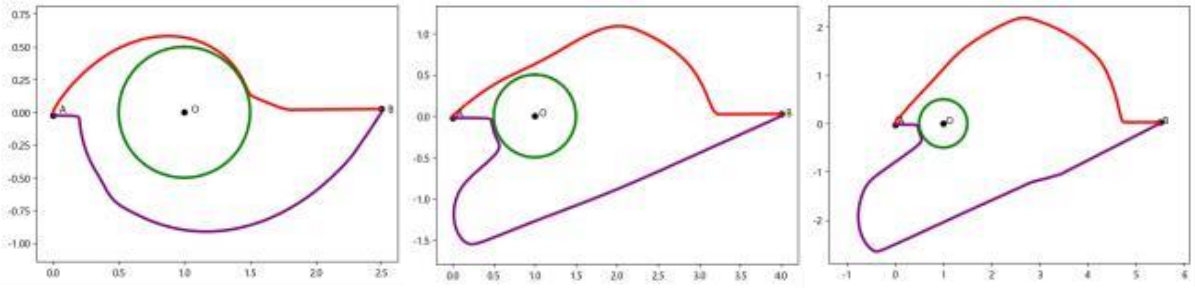


图 16 OB 长度对航迹的影响（左中右 OB 取值依次为 1.5, 3, 4.5, 单位：km）

通过分析，我们可以得出结论：OB 路径长度的变化会显著影响无人机的飞行轨迹。随着 OA 路径长度的增加，无人机飞行轨迹与障碍圆的偏离程度也会增大，特别是从站点 A 出发的无人机在后方的绕行路径会更长。进一步地，我们推导出路径长度与总飞行时间之间的关系，并将其示意图如下所示。

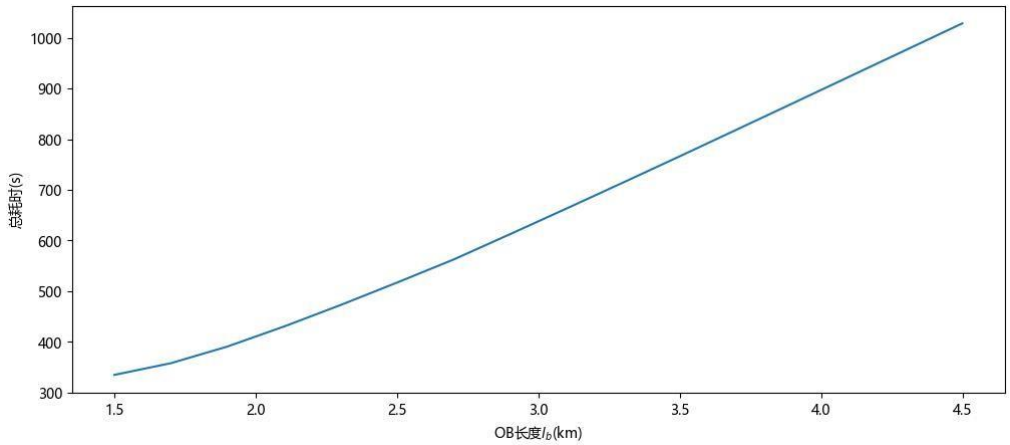



图 16 OB 长度对应总时长

经过分析，当 B 和圆心的距离的距离在 1.5km 的到 4.5km 之间，问题二最优轨迹长度总体依旧还是表现为近似的线性关系。

# 8 问题四求解

问题四考虑了当 2 号无人机的恒定速率在区间 [10, 30] m/s 内变化时，问题一中的飞行路径的变化情况。在问题一的模型中，我们考虑了 2 号无人机直接按照最短路径飞行，而 1 号无人机通过绕圈的方式在不碰面的“安全地带”进行避让。因此，如果 2 号无人机的恒定速率增加，将仅减少 1 号无人机的避让时间。1 号无人机到达目的地的时间变化函数 $T(x)$ 随着站点 B 到圆心的距离 $x$ 和速率 $v$ 的变化关系如下：

$$T(x) = \frac{4663.60}{v}$$

此外，在考虑问题二中的最优航迹在 2 号无人机的恒定速率变化时的情况时，由于速率的变化，存在一个临界节点。当 2 号无人机的速率超过该节点速率后，1 号无人机在到达点 E 前，2 号无人机已经飞行到一个安全点，因此两架无人机按照正常的航行路线不会发生碰撞。通过对问题二建立的模型进行求解，得到这个临界节点速率为 $19\,\mathrm{m/s}$。

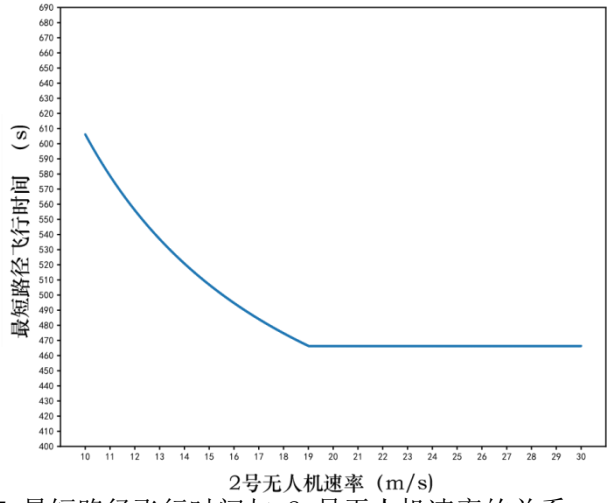



图 17 最短路径飞行时间与 2 号无人机速率的关系

从上图中可以得知，当 2 号无人机的速率大于 $19m/s$后，2 号到达目的站点的无人机最少用时变成了恒定的 $466.36s$，表明两架无人机均不用避让可以直接按最短飞行路径飞往目的地。

# 9 问题五模型的建立和求解

问题五考虑了当 2 号无人机的恒定速率在区间［10, 50］m/s 内变化，同时 B 站点到圆心的距离在区间［1, 10］km 内变化（其他参数保持不变）时，问题二中的最优航迹将如何改变。在这种情况下，需要同时考虑 2 号无人机的速率和 B 站点到圆心的距离两个因素的变化。

通过对问题二中模型的求解，我们可以得知当 B 站点到圆心的距离变化时，影响两无人机恰好相遇时的位置。另一方面，当 2 号无人机的速率变化时，其在问题二中横轴方向上的速度分量 $v_x$ 也会发生变化，这会影响 2 号无人机的从动点 $V_1$ 在追赶过程中追上 1 号无人机的运动点 $V_2$ 时的横坐标位置。

因此，B 站点到圆心的距离和 2 号无人机的速率同时影响着追逐问题中的追赶点位置，从而也影响了 1 号无人机是否在起点 A 处绕圈避让，以及 2 号无人机是否需要避让的情况。

实际上，每个无人机速率都对应一个 B 站点到圆心的距离值，使得在这种情况下 2 号无人机到达目的站点的时间达到最小。两架无人机的飞行情况可分为三种情况：

1. 两架无人机均无需避让，直接按照最短路径飞行；
2. 1 号无人机绕圈避让，2 号无人机先到达目的站点；
3. 2 号无人机绕圈避让，1 号无人机先到达终点。

通过绘制 2 号无人机速率、B 站点到圆心的距离以及 2 号无人机到达目的站点的最短时间之间的三维关系图，结果如下图所示：

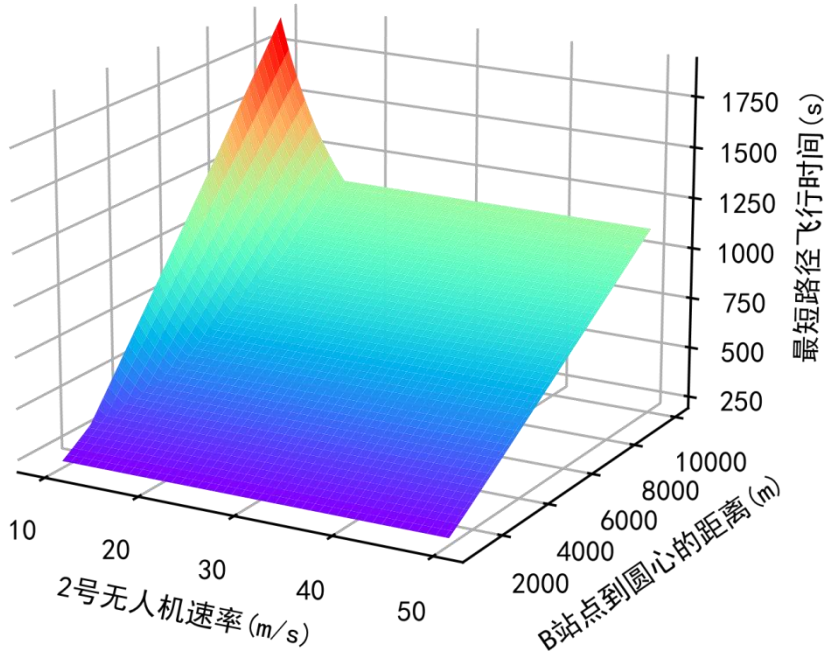
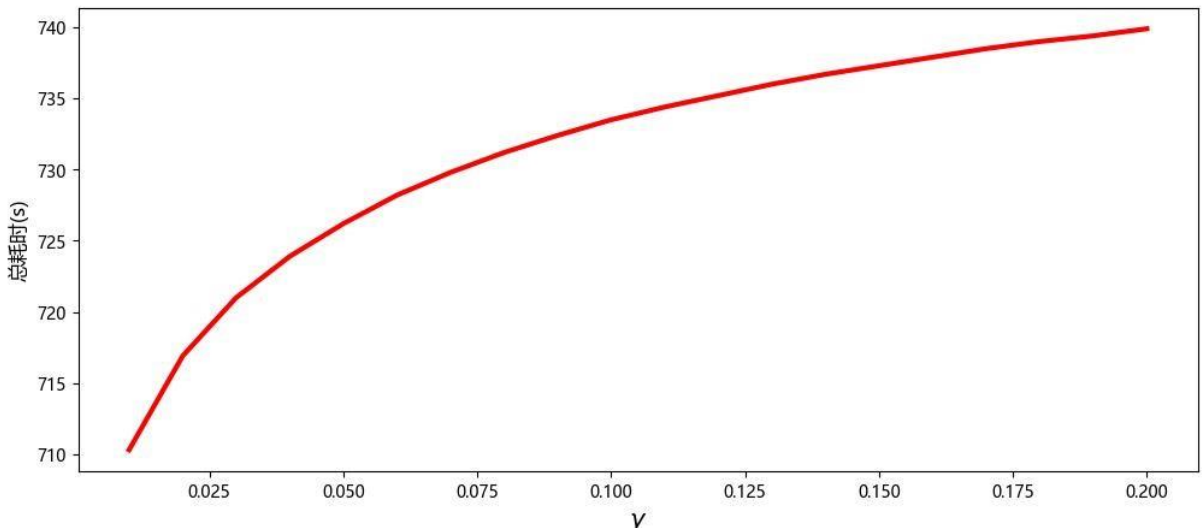



图 18 2 号无人机速率、B 站点到圆心的距离和 2 号无人机到达目的站点最短时间三者关系

通过上图可知，当 2 号无人机的速率越快，B 站点到圆心的距离越短时，2 号无人机到达目的站点的飞行时间缩短；当 2 号无人机的速率在 $10m/s$左右，B 站点到圆心的距离在后段区间变化时，2 号无人机到达目的站点的飞行时间总体大于其他情况且随着距离越大时间增加越快，这是由于此时 2 号无人机的速率变慢，且 B 站点到圆心的距离变长导致出现 1 号无人机需要绕圈避让的情况，增加了 1 号无人机到达目的站点的时间，站点 B 到圆心距离越长需要绕圈避让的时间越多。

## 10 灵敏度分析

为了准确了解参数相对位置调节增益系数 $k_r$ 对结果的影响，并验证模型的鲁棒性，本文在问题二中对相对位置调节增益系数 $k_r$ 进行了灵敏度分析。在此分析中，我们考虑了问题二的情景，将引力场增益系数 $k_a$ 设定为 1，障碍圆最大影响距离 $R$ 设定为 2 km，采样时间间隔设定为 0.1 s。我们按照从小到大的原则逐步取不同的 $k_r$ 值，即从 0.01 到 0.2 区间，以使得模型在约束条件下得以满足。结果如下图所示：



从图中可以明显看出，随着相对位置调节增益系数 $k_r$ 的增加，两架无人机的总飞行时间逐渐增加，但增长的速率逐渐减缓。这一结果揭示了模型对于不同 $k_r$ 值的鲁棒性，即在一定范围内调节 $k_r$ 不会引发剧烈的飞行时间变化，表明该模型在一定程度上具备稳健性。

# 11、模型的评价与推广

## 11.1 模型的优点

1.较稳定的几何模型求解方案：本文采用的无人机飞行几何模型能够提供稳定且可靠的最优解决方案。无论速度或站点距离的变化，路径的整体形态都保持不变，确保了飞行的稳定性和可预测性。

2.平滑且适用广泛的人工势场法：人工势场法为路径规划提供了平滑而合理的航迹，适用于各种不同形状和数量的障碍物。此外，该方法具备很强的灵活性，能够轻松扩展至三维空间，为解决多样化的避障问题提供了有效手段。

3.创新的相对位置调节场：引入人工势场法中的相对位置调节场，创新性地解决了无人机连线与障碍圆之间的约束问题。这个巧妙的调节场确保了无人机之间相对位置的精确控制，提升了避障方案的安全性和实用性。

## 11.2 模型的缺点

1.局部航迹规划限制： 本文所提出的改进人工势场法虽然在保证算法实现时效性方面具有优势，但并不能保证获得全局最优解。这一局限性可能会影响路径规划的最优性和完备性。

2. 潜在的局部最优和“死锁”问题：在人工势场法中，存在物体陷入局部最优解或震荡，甚至发生“死锁”现象的风险。这种情况可能影响路径规划的准确性和稳定性，需要进一步研究和改进。

3. 缺少全局规划算法结合： 尽管本文提出的模型在局部航迹规划方面取得了一定的成就，但仍未与全局规划算法进行深入结合，以获得更全面的路径规划结果。结合全局规划算法有助于寻找更优的航迹。

## 11.3 模型的推广

在面对多无人机和多障碍物的复杂环境时，可以对人工势场模型进行调整，以适应不同情境。例如，在多无人机飞行情况下，可为每架无人机添加以自身为源的斥力场，预防无人机之间的碰撞。这样的改进有助于确保无人机之间的安全间隔，提升整体的避障效果。在三维空间内，只需将力的梯度方向在垂直方向上扩展，即可将人工势场模型从二维拓展到三维。这种调整使得无人机在空中飞行时能够更准确地避开三维障碍物，保障飞行安全。

# 12、参考文献


[1] Dale A. Lawrence, Eric W. Frew and William J. Pisano. Lyapunov Vector Fields for Autonomous Unmanned Aircraft Flight Control [J]. Journal of Guidance, Control, and Dynamics, 2008:120-128.

[2] 薛申芳,刘长修,石爱芳.切线段与弧的长度关系[J].邢台师范高专学报,2001(04):46-47.

[3] Li J, Huang Y, Xu Z, et al. Path planning of UAV based on hierarchical genetic algorithm with optimized search region[C]. IEEE International Conference on Control & Automation.

IEEE, 2017: 33-38.

[4] 甄然, 甄士博, 吴学礼. 一种基于人工势场的无人机航迹规划算法[J]. 河北科技大学学报, 2017, 38(3): 278-284.

[5] 石为人, 黄兴华, 周伟. 基于改进人工势场法的移动机器人路径规划[[J]. 计算机应用, 2010, (08): 2021-2023.

[6] 江杰, 任恒靓. 基于改进人工势场法的移动机器人路径规划的研究[J]. 自动化应用, 2017(8).

[7] 陈守凤. 基于改进人工势场法的多无人机协同航迹规划算法研究[D].哈尔滨工业大学,2019.

[8] Simon Schopferer; Thorsten Pfeifer, Performance-aware flight path planning for unmanned aircraft in uniform wind fields[C]. IEEE International Conference on Unmanned Aircraft Systems (ICUAS), 2015:

## 13. 附录

**A. 支撑代码（ 见附录文件，使用 python 以及 matplotlib、pandas、numpy 库 ）**

代码文件：

第二问代码文件 question_2.ipynb
第三问代码文件 question_3.ipynb
第四问代码文件 question_3.ipynb
第五问代码文件 question_5.ipynb
灵敏度分析文件 sensitivity.ipynb

**部分问题代码（全代码见代码文件）：**

**问题二代码文件：见文件** question_2.ipynb,

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

# 参数设置
oa = 1
ob = 3.5
v = 0.01
R = 0.5
r = 0.03
alpha = 1
beta = 0.3
d0 = 2
gamma = 0.05

# 设置中文和负号正常显示
plt.rcParams['font.sans-serif'] = 'Microsoft YaHei'
plt.rcParams['axes.unicode_minus'] = False

# 距离函数
def distance(x,y):
    return np.sqrt([(x[0]-y[0])**2+(x[1]-y[1])**2])[0]

# 目标引力场函数(输入无人机坐标，目标坐标，增益系数)
def yin(wu,tar,alpha = alpha):
    F = alpha
    x = F*(tar[0]-wu[0])/distance(wu,tar)
    y = F*(tar[1]-wu[1])/distance(wu,tar)
    return np.array([x,y])
```

```
# 障碍圆斥力场函数(输入无人机坐标，障碍圆圆心坐标，增益系数，障碍圆最大影响距离)
def chi(wu,zhang,beta = beta,d0=d0):
    if distance(wu,zhang) > d0:
        return np.array([0,0])
    else:
        F = beta*(1/distance(wu,zhang) - 1/d0) / (distance(wu,zhang)**2)
        x = -F*(zhang[0]-wu[0])/distance(wu,zhang)
        y = -F*(zhang[1]-wu[1])/distance(wu,zhang)
        return np.array([x,y])

# 相对位置调节场函数(输入两无人机坐标，障碍圆圆心坐标，增益系数，障碍圆半径)
def adj(wu1,wu2,zhang,gamma = gamma,R = R):
    l1 = distance(wu1,wu2)
    l2 = distance(wu1,zhang)
    l3 = distance(wu2,zhang)
    cos = (l1**2+l2**2-l3**2)/2/l1/l2
    tt = l2 *cos
    # 垂足坐标
    chuix = (wu2[0]-wu1[0])*tt/l1 + wu1[0]
    chuiy = (wu2[1]-wu1[1])*tt/l1 + wu1[1]
    chui = [chuix, chuiy]
    # 圆心到连线距离
    D = distance(zhang,chui)
    if D > R/2:
        F = gamma*(2/R-1/(R-D))/((R-D)**2)
        x = F*(chuix - zhang[0])/distance(zhang,chui)
        y = F*(chuiy - zhang[1])/distance(zhang,chui)
        return np.array([x,y])
    else:
        return np.array([0,0])

judge = 0
beta = 0.01
dis = 0
minR = []
betalist = []
while judge < R:
    beta = beta + 0.01
    judge = 0
    # 初始化
    x1 = [] #A 出发无人机轨迹(x)
    y1 = [] #A 出发无人机轨迹(y)
    x2 = [] #B 出发无人机轨迹(x)
    y2 = [] #B 出发无人机轨迹(y)
    lo1 = [0,-0.025] #A 出发无人机实时位置
    lo2 = [oa+ob, 0.025] #B 出发无人机实时位置
    x1.append(lo1[0])
    y1.append(lo1[1])
```

```
    x2.append(lo2[0])
    y2.append(lo2[1])
    tt = 0.1 #时间间隔
    dx = tt * v #距离间隔
    a = [0,0]
    b = [oa+ob,0]
    zhang = [oa,0]

    while True:
        pp = adj(lo1,lo2,zhang)
        pp = np.array(-pp[1],pp[0])
        add1 = yin(lo1,b) + 5*chi(lo1,zhang,beta) + 0.4*pp +
0.1*adj(lo1,lo2,zhang)
        lo1[0] = lo1[0] + dx*add1[0]/math.sqrt(add1[0]**2+add1[1]**2)
        lo1[1] = lo1[1] + dx*add1[1]/math.sqrt(add1[0]**2+add1[1]**2)
        add2 = yin(lo2,a) + chi(lo2,zhang,beta) + adj(lo2,lo1,zhang)
        lo2[0] = lo2[0] + dx*add2[0]/math.sqrt(add2[0]**2+add2[1]**2)
        lo2[1] = lo2[1] + dx*add2[1]/math.sqrt(add2[0]**2+add2[1]**2)
        x1.append(lo1[0])
        y1.append(lo1[1])
        x2.append(lo2[0])
        y2.append(lo2[1])
        if distance(lo1,b)<0.001 or distance(lo2,a)<0.001:
            break

    if distance(lo1,b) > 0.001:
        while distance(lo1,b) > 0.001:
            add1 = yin(lo1,b) + chi(lo1,zhang,beta)
            lo1[0] = lo1[0] + dx*add1[0]/math.sqrt(add1[0]**2+add1[1]**2)
            lo1[1] = lo1[1] + dx*add1[1]/math.sqrt(add1[0]**2+add1[1]**2)
            x1.append(lo1[0])
            y1.append(lo1[1])
    else:
        while distance(lo2,a) > 0.001:
            add2 = yin(lo2,a) + chi(lo2,zhang,beta)
            lo2[0] = lo2[0] + dx*add2[0]/math.sqrt(add2[0]**2+add2[1]**2)
            lo2[1] = lo2[1] + dx*add2[1]/math.sqrt(add2[0]**2+add2[1]**2)
            x2.append(lo2[0])
            y2.append(lo2[1])
    d1 = []
    for i,j in enumerate(x1):
        d1.append(distance([x1[i],y1[i]],zhang))
    for i,j in enumerate(x2):
        d1.append(distance([x2[i],y2[i]],zhang))
    judge = min(d1)
    minR.append(judge)
    betalist.append(beta)

print('beta:',beta)
```

```
print('A 出发的无人机耗时',tt*len(x1),'s')
print('B 出发的无人机耗时',tt*len(x2),'s')
plt.figure(figsize=(12, 5))
plt.scatter([0,oa,oa+ob], [-0.025,0,0.025], c="black")
plt.annotate('A', xy=(0, 0), xytext=(0.05, 0))
plt.annotate('O', xy=(0, 0), xytext=(0.05+oa, 0))
plt.annotate('B', xy=(0, 0), xytext=(0.05+oa+ob, 0))
plt.plot(x1,y1,c='purple',linewidth=3)
plt.plot(x2,y2,c='red',linewidth=3)
xx = [oa+R*math.sin(i/1000*2*math.pi) for i in range(1000)]
yy = [R*math.cos(i/1000*2*math.pi) for i in range(1000)]
plt.plot(xx,yy,c='green',linewidth=3)
plt.axis('equal')
```

**问题三代码文件：见文件 question_3.ipynb，**

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

# 参数设置
oa = 1
ob = 3.5
v = 0.01
R = 0.5
r = 0.03
alpha = 1
beta = 0.3
d0 = 2
gamma = 0.05

# 设置中文和负号正常显示
plt.rcParams['font.sans-serif'] = 'Microsoft YaHei'
plt.rcParams['axes.unicode_minus'] = False

# 距离函数
def distance(x,y):
    return np.sqrt([(x[0]-y[0])**2+(x[1]-y[1])**2])[0]

# 目标引力场函数(输入无人机坐标，目标坐标，增益系数)
def yin(wu,tar,alpha = alpha):
    F = alpha
    x = F*(tar[0]-wu[0])/distance(wu,tar)
```

```
    y = F*(tar[1]-wu[1])/distance(wu,tar)
    return np.array([x,y])

# 障碍圆斥力场函数(输入无人机坐标，障碍圆圆心坐标，增益系数，障碍圆最大影响距离)
def chi(wu,zhang,beta = beta,d0=d0):
    if distance(wu,zhang) > d0:
        return np.array([0,0])
    else:
        F = beta*(1/distance(wu,zhang) - 1/d0) / (distance(wu,zhang)**2)
        x = -F*(zhang[0]-wu[0])/distance(wu,zhang)
        y = -F*(zhang[1]-wu[1])/distance(wu,zhang)
        return np.array([x,y])

# 相对位置调节场函数(输入两无人机坐标，障碍圆圆心坐标，增益系数，障碍圆半径)
def adj(wu1,wu2,zhang,gamma = gamma,R = R):
    l1 = distance(wu1,wu2)
    l2 = distance(wu1,zhang)
    l3 = distance(wu2,zhang)
    cos = (l1**2+l2**2-l3**2)/2/l1/l2
    tt = l2 *cos
    # 垂足坐标
    chuix = (wu2[0]-wu1[0])*tt/l1 + wu1[0]
    chuiy = (wu2[1]-wu1[1])*tt/l1 + wu1[1]
    chui = [chuix, chuiy]
    # 圆心到连线距离
    D = distance(zhang,chui)
    if D > R/2:
        F = gamma*(2/R-1/(R-D))/((R-D)**2)
        x = F*(chuix - zhang[0])/distance(zhang,chui)
        y = F*(chuiy - zhang[1])/distance(zhang,chui)
        return np.array([x,y])
    else:
        return np.array([0,0])
haoshi = []
for ii in range(16):
    ob = 1.5+0.2*ii
    judge = 0
    beta = 0.01
    dis = 0
    while judge < R:
        beta = beta + 0.01
        judge = 0
        # 初始化
        x1 = [] #A 出发无人机轨迹(x)
        y1 = [] #A 出发无人机轨迹(y)
        x2 = [] #B 出发无人机轨迹(x)
        y2 = [] #B 出发无人机轨迹(y)
        lo1 = [0,-0.025] #A 出发无人机实时位置
        lo2 = [oa+ob, 0.025] #B 出发无人机实时位置
```

```
        x1.append(lo1[0])
        y1.append(lo1[1])
        x2.append(lo2[0])
        y2.append(lo2[1])
        tt = 0.1 #时间间隔
        dx = tt * v #距离间隔
        a = [0,0]
        b = [oa+ob,0]
        zhang = [oa,0]

        while True:
            pp = adj(lo1,lo2,zhang)
            pp = np.array(-pp[1],pp[0])
            add1 = yin(lo1,b) + 5*chi(lo1,zhang,beta) + 0.4*pp +
0.1*adj(lo1,lo2,zhang)
            lo1[0] = lo1[0] + dx*add1[0]/math.sqrt(add1[0]**2+add1[1]**2)
            lo1[1] = lo1[1] + dx*add1[1]/math.sqrt(add1[0]**2+add1[1]**2)
            add2 = yin(lo2,a) + chi(lo2,zhang,beta) + adj(lo2,lo1,zhang)
            lo2[0] = lo2[0] + dx*add2[0]/math.sqrt(add2[0]**2+add2[1]**2)
            lo2[1] = lo2[1] + dx*add2[1]/math.sqrt(add2[0]**2+add2[1]**2)
            x1.append(lo1[0])
            y1.append(lo1[1])
            x2.append(lo2[0])
            y2.append(lo2[1])
            if distance(lo1,b)<0.001 or distance(lo2,a)<0.001:
                break

        if distance(lo1,b) > 0.001:
            while distance(lo1,b) > 0.001:
                add1 = yin(lo1,b) + chi(lo1,zhang,beta)
                lo1[0] = lo1[0] + dx*add1[0]/math.sqrt(add1[0]**2+add1[1]**2)
                lo1[1] = lo1[1] + dx*add1[1]/math.sqrt(add1[0]**2+add1[1]**2)
                x1.append(lo1[0])
                y1.append(lo1[1])
        else:
            while distance(lo2,a) > 0.001:
                add2 = yin(lo2,a) + chi(lo2,zhang,beta)
                lo2[0] = lo2[0] + dx*add2[0]/math.sqrt(add2[0]**2+add2[1]**2)
                lo2[1] = lo2[1] + dx*add2[1]/math.sqrt(add2[0]**2+add2[1]**2)
                x2.append(lo2[0])
                y2.append(lo2[1])
        d1 = []
        for i,j in enumerate(x1):
            d1.append(distance([x1[i],y1[i]],zhang))
        for i,j in enumerate(x2):
            d1.append(distance([x2[i],y2[i]],zhang))
        judge = min(d1)
    haoshi.append(max(tt*len(x1),tt*len(x2)))
```

**问题三代码文件：见文件 question_3.ipynb，**

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

# 参数设置
oa = 1
ob = 3.5
v = 0.01
R = 0.5
r = 0.03
alpha = 1
beta = 0.3
d0 = 2
gamma = 0.05

# 设置中文和负号正常显示
plt.rcParams['font.sans-serif'] = 'Microsoft YaHei'
plt.rcParams['axes.unicode_minus'] = False

# 距离函数
def distance(x,y):
    return np.sqrt([(x[0]-y[0])**2+(x[1]-y[1])**2])[0]

# 目标引力场函数(输入无人机坐标，目标坐标，增益系数)
def yin(wu,tar,alpha = alpha):
    F = alpha
    x = F*(tar[0]-wu[0])/distance(wu,tar)
    y = F*(tar[1]-wu[1])/distance(wu,tar)
    return np.array([x,y])

# 障碍圆斥力场函数(输入无人机坐标，障碍圆圆心坐标，增益系数，障碍圆最大影响距离)
def chi(wu,zhang,beta = beta,d0=d0):
    if distance(wu,zhang) > d0:
        return np.array([0,0])
    else:
        F = beta*(1/distance(wu,zhang) - 1/d0) / (distance(wu,zhang)**2)
        x = -F*(zhang[0]-wu[0])/distance(wu,zhang)
        y = -F*(zhang[1]-wu[1])/distance(wu,zhang)
        return np.array([x,y])

# 相对位置调节场函数(输入两无人机坐标，障碍圆圆心坐标，增益系数，障碍圆半径)
def adj(wu1,wu2,zhang,gamma = gamma,R = R):
    l1 = distance(wu1,wu2)
    l2 = distance(wu1,zhang)
```

```
    l3 = distance(wu2,zhang)
    cos = (l1**2+l2**2-l3**2)/2/l1/l2
    tt = l2 *cos
    # 垂足坐标
    chuix = (wu2[0]-wu1[0])*tt/l1 + wu1[0]
    chuiy = (wu2[1]-wu1[1])*tt/l1 + wu1[1]
    chui = [chuix, chuiy]
    # 圆心到连线距离
    D = distance(zhang,chui)
    if D > R/2:
        F = gamma*(2/R-1/(R-D))/((R-D)**2)
        x = F*(chuix - zhang[0])/distance(zhang,chui)
        y = F*(chuiy - zhang[1])/distance(zhang,chui)
        return np.array([x,y])
    else:
        return np.array([0,0])
haoshi = []
for ii in range(16):
    ob = 1.5+0.2*ii
    judge = 0
    beta = 0.01
    dis = 0
    while judge < R:
        beta = beta + 0.01
        judge = 0
        # 初始化
        x1 = [] #A 出发无人机轨迹(x)
        y1 = [] #A 出发无人机轨迹(y)
        x2 = [] #B 出发无人机轨迹(x)
        y2 = [] #B 出发无人机轨迹(y)
        lo1 = [0,-0.025] #A 出发无人机实时位置
        lo2 = [oa+ob, 0.025] #B 出发无人机实时位置
        x1.append(lo1[0])
        y1.append(lo1[1])
        x2.append(lo2[0])
        y2.append(lo2[1])
        tt = 0.1 #时间间隔
        dx = tt * v #距离间隔
        a = [0,0]
        b = [oa+ob,0]
        zhang = [oa,0]

        while True:
            pp = adj(lo1,lo2,zhang)
            pp = np.array(-pp[1],pp[0])
            add1 = yin(lo1,b) + 5*chi(lo1,zhang,beta) + 0.4*pp +
0.1*adj(lo1,lo2,zhang)
            lo1[0] = lo1[0] + dx*add1[0]/math.sqrt(add1[0]**2+add1[1]**2)
            lo1[1] = lo1[1] + dx*add1[1]/math.sqrt(add1[0]**2+add1[1]**2)
```

```
            add2 = yin(lo2,a) + chi(lo2,zhang,beta) + adj(lo2,lo1,zhang)
            lo2[0] = lo2[0] + dx*add2[0]/math.sqrt(add2[0]**2+add2[1]**2)
            lo2[1] = lo2[1] + dx*add2[1]/math.sqrt(add2[0]**2+add2[1]**2)
            x1.append(lo1[0])
            y1.append(lo1[1])
            x2.append(lo2[0])
            y2.append(lo2[1])
            if distance(lo1,b)<0.001 or distance(lo2,a)<0.001:
                break

        if distance(lo1,b) > 0.001:
            while distance(lo1,b) > 0.001:
                add1 = yin(lo1,b) + chi(lo1,zhang,beta)
                lo1[0] = lo1[0] + dx*add1[0]/math.sqrt(add1[0]**2+add1[1]**2)
                lo1[1] = lo1[1] + dx*add1[1]/math.sqrt(add1[0]**2+add1[1]**2)
                x1.append(lo1[0])
                y1.append(lo1[1])
        else:
            while distance(lo2,a) > 0.001:
                add2 = yin(lo2,a) + chi(lo2,zhang,beta)
                lo2[0] = lo2[0] + dx*add2[0]/math.sqrt(add2[0]**2+add2[1]**2)
                lo2[1] = lo2[1] + dx*add2[1]/math.sqrt(add2[0]**2+add2[1]**2)
                x2.append(lo2[0])
                y2.append(lo2[1])
        d1 = []
        for i,j in enumerate(x1):
            d1.append(distance([x1[i],y1[i]],zhang))
        for i,j in enumerate(x2):
            d1.append(distance([x2[i],y2[i]],zhang))
        judge = min(d1)
    haoshi.append(max(tt*len(x1),tt*len(x2)))
```