中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生数学建模竞赛

学　　校	华中科技大学
参赛队号	**20104870033**
队员姓名	**1.张　星**
	2.胡高阳
	3.杨　博

中国研究生创新实践系列大赛
“华为杯”第十七届中国研究生
数学建模竞赛

题 目 无人机集群协同对抗

摘 要：

随着智能飞行器的迅速发展，无人机集群作战成为空战的发展趋势之一。无人机集群作战研究包括无人机编队飞行、航迹规划和分布式协同对抗策略等几个方面。本文研究二维空间平行边界内的蓝方无人机单方向突防与红方无人机集群防御策略，讨论了突防方与防御方的最优运动策略，并求出不同条件下的通道带宽的极大极小值。

针对问题一，考虑红方无人机集群对称布置的基础上，通过设定红方无人机与蓝方无人机的最优运动策略，对矩形运动区域内的所有坐标点进行判定，得出两个单波无人机集群条件下蓝方必能突防的初始位置区域。

针对问题二，考虑红方无人机集群对称布置的基础上，通过设定红方无人机与蓝方无人机的最优运动策略，求出题设条件下存在一个通道带宽 M 的下限 $M_{\min}=8.44km$，当实际通道带宽 M 比 $M_{\min}$ 大时，蓝方无人机一定能突破红方无人机集群的拦截。

针对问题三，在得到问题二中单波无人机通道宽带下限的基础上，通过计算蓝方无人机被拦截的临界状态，倒推出题设条件下存在一个通道带宽 M 的上限 $M_{\max}=16.52km$，实际通道带宽 小于 $M_{\max}$ 时，无论蓝方无人机采用什么样的突防策略，红方无人机集群一定能成功阻止蓝方无人机的突防。

针对多约束条件下红蓝双方无人机协同对抗问题，本文讨论了红蓝双方的最优运动策略，并建立模型求解计算。三个问题均在较短时间内解出。模型实际应用价值高，针对大规模和复杂度比较高的问题，模型求解还需要进一步研究。

关键词：无人机对抗 捕鱼策略 微分对策 博弈策略

目录

1. 问题重述

新一代人工智能技术和自主技术快速走向战场，将催生新型作战力量，颠覆传统战争模式，未来战争必将是智能化战争。无人机集群作战作为智能作战的重要形式，正在崭露头角。通过多架无人机协同侦察、协同探测、协同跟踪、协同攻击、协同拦截等，共同完成较复杂的作战任务。

现考虑红、蓝双方的无人机集群在平面区域内的协同对抗问题。蓝方作为进攻方，希望突破红方无人机的拦截，成功抵达目的地遂行军事行动；红方则希望在给定的区域内完成对蓝方无人机的拦截，阻止蓝方的突防。本对抗区域约定为图 1 所示的矩形区域 ABCD，攻击纵深即 BC 之间的距离为 $L = 50km$，蓝方无人机的飞行轨迹不能越过 AD、BC 两边，即考虑的是攻击通道（突防走廊）带宽有一个限定约束的情形，通道带宽即 AB 之间的距离记为 M。蓝方无人机的速度为 $V_E = 250m/s$，最小转弯半径为 $R_E = 500m$；红方无人机的速度为 $V_P = 200m/s$，最小转弯半径为 $R_P = 350m$。

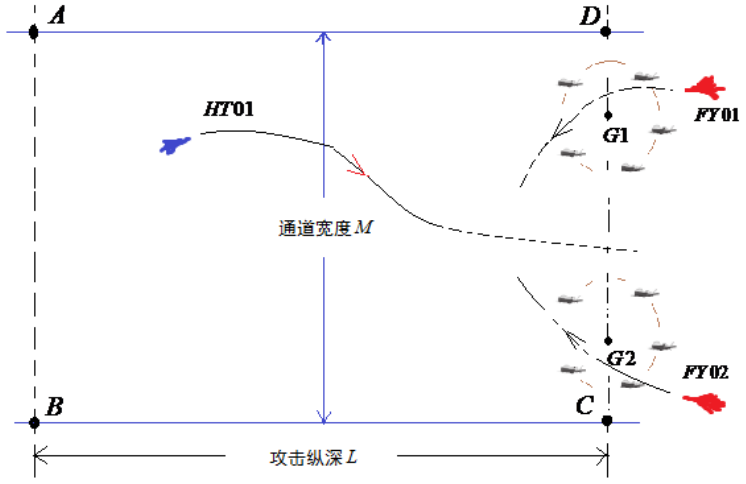

图 1　红方两个无人机集群拦截一架蓝方突防无人机示意图

鉴于蓝方无人机在机动速度上具有优势，红方考虑通过多无人机的协同，即通过数量上的优势部分弥补性能上的劣势，以提高己方的对抗效果。红方执行拦截任务的无人机由运载机携带至攻击位置，然后根据对抗需要发射一定数量的无人机，组成无人机集群与蓝方突防无人机对抗。如图 1 所示的对抗状态，红方为阻止蓝方无人机的突防，分别由运载机 FY01 和 FY02 各发射 5 架无人机组成两个无人机集群遂行拦截任务。当蓝方突防无人机与红方至少 2 架无人机的距离均小于 $R = 300m$ 时，就认为红方成功拦截了蓝方突防无人机。同时，根据任务要求，蓝方无人机需在 $360s$ 内越过边界 CD，否则视为突防失败。红蓝双方都想充分利用自身的优势，通过运用最优机动策略以实现突防与拦截的目的。

红方无人机集群可以根据实际需要进行编队飞行，红方无人机集群采用了其中的一种队形，5 架无人机位置近似分布在一个圆周上，任何相邻两架无人机的间距相同。为控制、通信以及相互避撞的需要，要求红方任何两架无人机的间距需大于 $30m$，每一架无人机与本集群中至少两架无人机的距离不超过 $200m$。红方运载机与所属无人机集群中至少一架无人机的距离不超过 $10km$，与任何一架无人机的距离需大于 $100m$，同时为安全需要，与

蓝方的突防无人机的距离需大于 $5km$。红方运载机的速度为 $V_{红}=300m/s$，转弯半径不小于 $1000m$。

问题假设和约束：

(1) 红蓝双方无人机的速度保持不变，运动的方向受最小转弯半径的限制。

(2) 所有无人机均位于同一高度，且红蓝双方均了解所有无人机的实时位置信息。

(3) 红方运载机和无人机的飞行轨迹不受 ABCD 边界的限制。

(4) 红方无人机集群初始时刻的位置位于一个圆周上均匀分布且瞬间布设完成，对抗过程可根据需要随时调整队形。

通过建立数学模型，研究下列问题：

问题 1：对抗伊始红方 2 个无人机集群的圆周中心分别位于 G1 和 G2，圆周半径为 $100m$，其中 $DG1=20km$，$G1G2=30km$，$CG2=20km$。建模分析蓝方无人机处于矩形区域 ABCD 内哪些位置时，无论红方无人机采用什么样的追击策略，蓝方无人机总能采用合适的策略以躲避红方的拦截，实现成功突防；讨论蓝方无人机相应的最优突防策略。

问题 2：对抗伊始蓝方突防无人机位于边界 AB 的中心点，红方 2 个无人机集群的圆周中心分别位于 G1 和 G2，圆周半径为 $100m$，其中 G1 和 G2 位于边界 CD 上，具体位置根据需要确定。试建模分析是否存在一个通道带宽 $M_{\min}$ 的下限 ，当实际通道带宽 M 比 $M_{\min}$ 大时，蓝方无人机一定能突破红方无人机集群的拦截；给出此种情形下蓝方无人机时间最短的突防策略。

问题 3：红方每架运载机可分两个波次共发射 10 架无人机，组成两个无人机集群遂行拦截任务，每个无人机集群的无人机数量不少于 3 架。每一波次发射时无人机集群初始队形如图 1 所示的圆周构型，运载机与圆周中心的距离为 $2km$，随后无人机集群的队形可根据需要调整，但要求满足相应的间距约束。对抗伊始，蓝方无人机位于边界 AB 的中心，通道带宽 $70km$；红方两架运载机分别位于边界 CD 上 G1 点和 G2 点，并开始发射第一波次的无人机集群，运载机和无人机集群中心具体位置根据需要确定。运载机第二波次发射无人机集群时，必须保证运载机与第一波次发射的无人机集群满足间距上的约束。讨论红方两架运载机两个波次发射的无人机数量、每架运载机第二波次发射的时刻和位置以及第二波次发射的无人机集群的中心位置，以实现最优的拦截效果；进一步具体建模分析是否存在一个通道带宽 M 的上限 $M_{\max}$，当实际通道带宽 M 小于 $M_{\max}$ 时，无论蓝方无人机采用什么样的突防策略，红方无人机集群均存在相应的拦截策略，在区域 ABCD 内成功阻止蓝方无人机的突防。

问题 4：通道带宽 $M=100km$，蓝方 3 架突防无人机组成突防集群从矩形边界 AB 一侧开始突防（任 2 架突防无人机的间距需大于 $100m$），红方 5 架运载机各携带 10 架无人机，从边界 CD 一侧同时开始遂行协同拦截任务。红方每架运载机分两个波次发射无人机，分别组成两个无人机集群，每个集群的无人机数量不少于 3 架；每架运载机第一波次发射无人机的时刻为初始对抗时刻，与所属无人机集群几何构型圆周中心的距离为 $2km$。红方运载机初始位置、红方运载机发射的第一个波次的无人机集群中心位置、红方运载机发射第二波次无人机集群的时刻和位置、第二波次发射的无人机集群中心位置、两个波次无人机数量以及蓝方突防无人机初始位置根据需要确定。蓝方希望尽可能多的无人机突防成功，红方则希望成功拦截尽可能多的蓝方无人机。试讨论红方最优拦截策略和蓝方最优突防策略。

2. 符号说明

序号	符号	符号说明
1	X_{r}	红色无人机 x 轴方向的坐标分量
2	$\dot{X}_{\mathrm{r}}$	红色无人机 x 轴方向的速度分量
3	X_{b}	蓝色无人机 x 轴方向的坐标分量
4	$\dot{X}_{\mathrm{b}}$	蓝色无人机 x 轴方向的速度分量
5	Y_{r}	红色无人机 y 轴方向的坐标分量
6	$\dot{Y}_{\mathrm{r}}$	红色无人机 y 轴方向的速度分量
7	Y_{b}	蓝色无人机 y 轴方向的坐标分量
8	$\dot{Y}_{\mathrm{b}}$	蓝色无人机 y 轴方向的坐标分量
9	α	红色无人机与 y 轴正方向的夹角
10	$\dot{\alpha}$	红色无人机转动加速度
11	β	蓝色无人机与 y 轴负方向的夹角
12	$\dot{\beta}$	蓝色无人机转动加速度
13	R_{r}	红色无人机转动半径
14	R_{b}	蓝色无人机转动半径
15	μ	红色无人机控制转动方向
16	v	蓝色无人机控制转动方向
17	l	捕获半径
18	Δt	时间间隔
19	r	红蓝无人机相对距离

3. 模型简化及假设

阿波罗尼圆广泛应用于追逃问题的分析[1-3]，其示意图如图 1 所示，这种方法也适用于无人机集群协同对抗问题，其由防御者（红色无人机）可能遇到入侵者（蓝色无人机）的点组成，阿波罗尼圆的变化取决于防御者（红色无人机）和入侵者（蓝色无人机）的运动速度以及当前位置，即$|AB| / V_r = |FB| / V_b$。为定量表征其位置关系，令入侵者坐标为（X_b, Y_b），令防御者坐标为（X_r, Y_r），$\varepsilon = V_r / V_b$。阿波罗尼圆的圆心可表示为:（$(X_r - \varepsilon^2 X_b) / (1 - \varepsilon^2)$, $(Y_r - \varepsilon^2 Y_b) / (1 - \varepsilon^2)$），其半径可表示为:

$$R = \frac{\varepsilon\sqrt{\left(X_r - X_b\right)^2 + \left(Y_r - Y_b\right)^2}}{1 - \varepsilon^2}$$

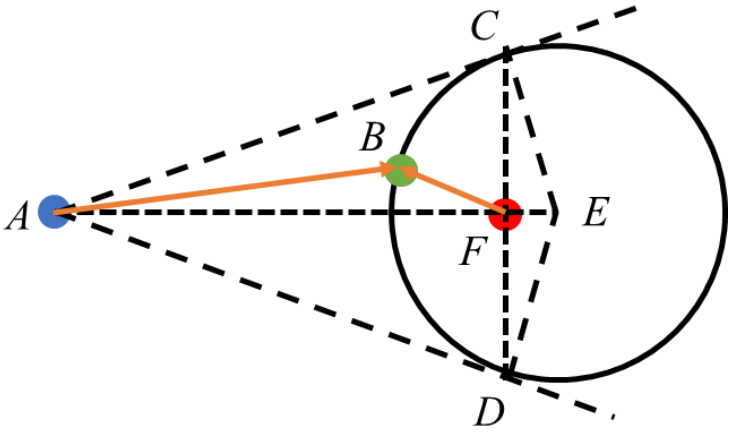

图 1 阿波罗尼圆示意图

微分对策，建立在动态微分状态下的最优对策，结合了博弈论与控制论能够有效地研究多目标的最优方案。微分对策的要素包括：参与者、对策、代价、局势等，其中对策中的均衡状态对应于微分对策中的鞍点[4]。本题中可以认为无人机协同对抗对于红色无人机群一种阵地防御问题，而对于蓝色无人机是一种生存线对策，如图 2 所示。由于平面关于水平中轴对称，取半对称空间，令右边界为 x 轴，入侵者和防御者位于 x 轴的上半平面，入侵者坐标为（X_b, Y_b），防御者坐标为（X_r, Y_r），则存在如下的运动状态方程组：

$$\begin{cases} \dot{X}_r = V_r \sin\alpha \\ \dot{X}_b = V_b \sin\beta \\ \dot{Y}_r = V_r \cos\alpha \\ \dot{Y}_b = -V_b \cos\beta \\ \dot{\alpha} = \dfrac{V_r \mu}{R_r} \\ \dot{\beta} = \dfrac{V_b \nu}{R_b} \end{cases}$$

式中，$\dot{X}_r$ 为红色无人机在 x 轴方向绝对速度，$\dot{X}_b$ 为蓝色无人机在 x 轴方向绝对速度，

V_r 为红色无人机线速度，V_b 为蓝色无人机线速度，α 为红色无人机与 y 轴正方向的夹角，β 为蓝色无人机与 y 轴负方向的夹角，$\dot{Y}_r$ 为红色无人机在 y 轴方向绝对速度，$\dot{Y}_b$ 为蓝色无人机在 y 轴方向绝对速度，$\dot{\alpha}$ 红色无人机转动加速度，$\dot{\beta}$ 蓝色无人机转动加速度，R_r 为红色无人机转动半径，R_b 为蓝色无人机转动半径，μ 为红色无人机控制转动方向，v 为蓝色无人机控制转动方向，（-1 <= μ、v<= 1）。

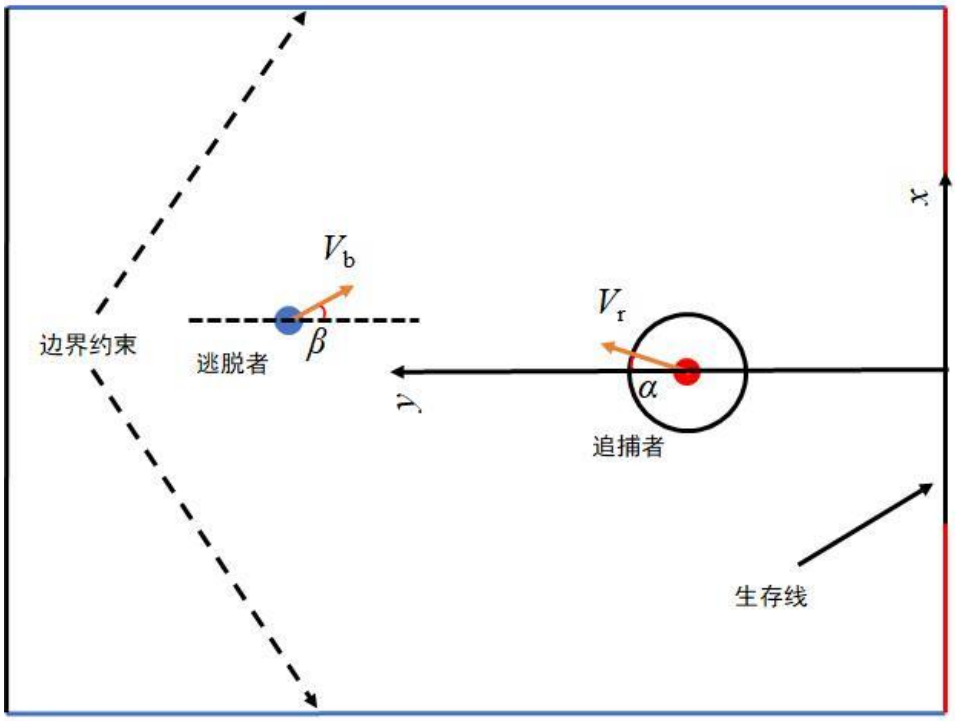

图 2 本题生存对策问题示意图

记哈密顿函数为：

$$H = \lambda_1\dot{X}_r + \lambda_2\dot{X}_b + \lambda_3\dot{Y}_r + \lambda_4\dot{Y}_b + \lambda_5\dot{\alpha} + \lambda_6\dot{\beta}$$

带入运动状态方程：

$$H = V_r(\lambda_1 sina + \lambda_3 cosa + \lambda_5 \frac{\mu}{R_r}) + V_b(\lambda_2 sin\beta - \lambda_4 cos\beta + \lambda_6 \frac{v}{R_b})$$

其伴随方程：

$$\dot{\lambda}_1 = 0 \quad \dot{\lambda}_2 = 0 \quad \dot{\lambda}_3 = 0 \quad \dot{\lambda}_4 = 0$$
$$\dot{\lambda}_5 = \lambda_1 cosa - \lambda_3 sina \quad \dot{\lambda}_6 = \lambda_2 cos\beta + \lambda_4 sin\beta$$

其边界约束：

$$\wp = （X_b、Y_b、X_r、Y_r | -15<=X_b<=20 \ \& \ Y_b=0 \ \& \ (X_b-Y_b)^2+（X_r+Y_r)^2<=1^2 \ ）$$

得到第一个基本方程：

$$\min_{\mu}\max_{v}\left[V_r(\lambda_1 sina + \lambda_3 cosa + \lambda_5 \frac{\mu}{R_r}) + V_b(\lambda_2 sin\beta - \lambda_4 cos\beta + \lambda_6 \frac{v}{R_b})\right] = 0$$

解基本方程：

$$\lambda_1 cos\alpha - \lambda_3 sin\alpha + \lambda_5 = 0$$

$$\lambda_2 cos\beta + \lambda_4 sin\beta + \lambda_6 = 0$$

3.1 简化 1

本文将红方无人机集群编队的最大拦截范围简化为一个圆，简化过程如图 3 所示。由于红方无人机集群的初始队形为一个半径 $100m$ 的圆周均匀分布，因此每架红方无人机的初始位置可为该圆周的任意位置。在对抗过程中，红方无人机若保持圆周阵型，其相对位置可任意调节。当至少两架红方无人机与蓝方无人机的距离小于 $300m$ 时，蓝方无人机被拦截。因此两个半径 $300m$ 的圆的交点的意义即为拦截的极限范围值。当两架红方无人机的距离为 $30m$ 时，其极限范围值达到最大。综上所述，可将无人机集群的最大拦截范围简化为一个半径为 $398m$ 的圆。

在后续的问题求解中，都假设红方无人机集群整体为一个点，当蓝方无人机与该点的范围小于 398m 时，蓝方无人机被拦截[5]。

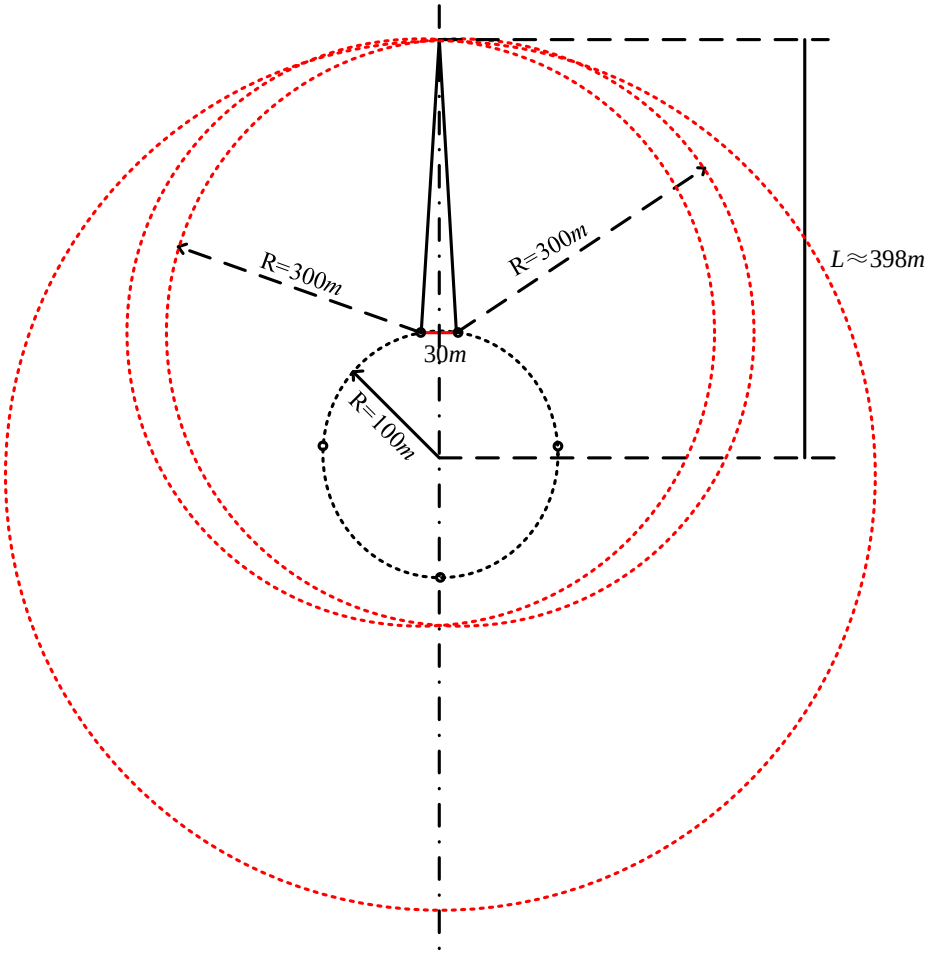

图 3 无人机集群简化示意图

3.2 简化 2

本文的所有问题均可简化为捕鱼对策(Fishing Game)[6-8]，如图 4 所示，具体来说，两个追捕者与一个逃脱者在一个平面限定区域内中相向运动，当逃脱者成功地从两个追捕者的一侧运动到另一侧，且与追捕者的最小距离一直大于捕获半径时，逃脱者胜利；否则，追捕者胜利。如图 5 所示，圆 Q_1 和圆 Q_2 分别为 P_1 和 P_2 相对于 E 的阿波罗尼斯圆，EA 和 EB 分别为 E 点到两圆的切线。当切线 EA 与 EB 之间存在空隙，逃脱者只要在此空隙间运动且在追捕者的捕获半径以外，则追捕者 P_1 和 P_2 就不能捕获到逃脱者。查文中通过求解连续时间捕鱼微分对策，得出在基于相对距离的目标对策上，**逃脱者的最优策略为不断调整速度指向两个追捕者连线的中点，而追捕者的最优策略为不断调整速度方向垂直于逃脱者的速度方向[5]。**本文所有问题中的红蓝方无人机的最优运动策略均基于此。

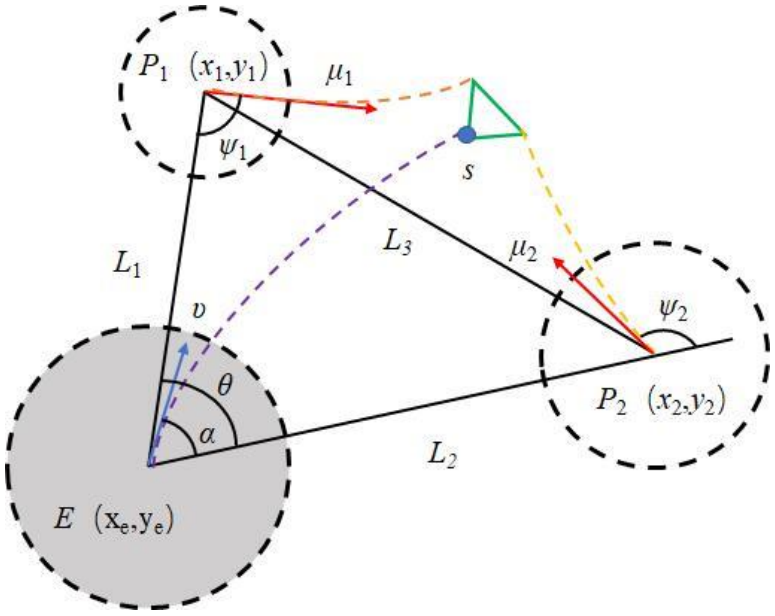

图 4 固定坐标系下的捕鱼对策模型

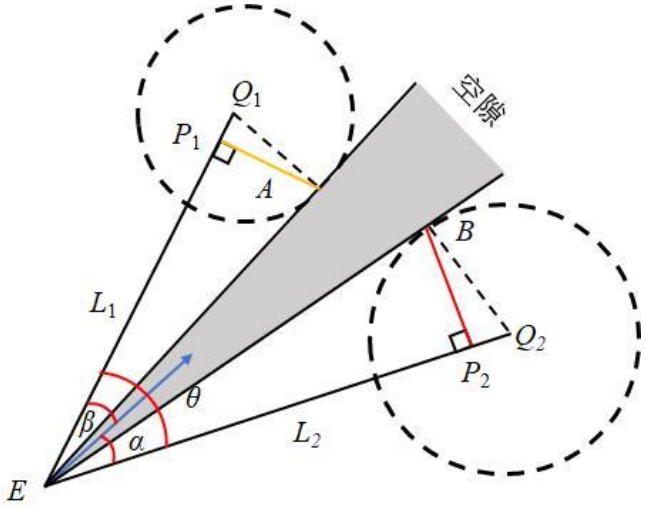

图 5 逃脱者的逃脱条件

3.3 简化 3

将上下平行边界等效为一个红方无人机集群，其只能沿着边界的方向移动，水平位置与最近的一个红方无人机集群时刻保持一致，拦截半径为 0。因此可与最近的一个红方无人机集群形成一个“渔网”，此时**蓝方无人机的最优突防策略为不断调整速度方向指向红方无人机与运动边界垂线的中点。**如图 6 所示。

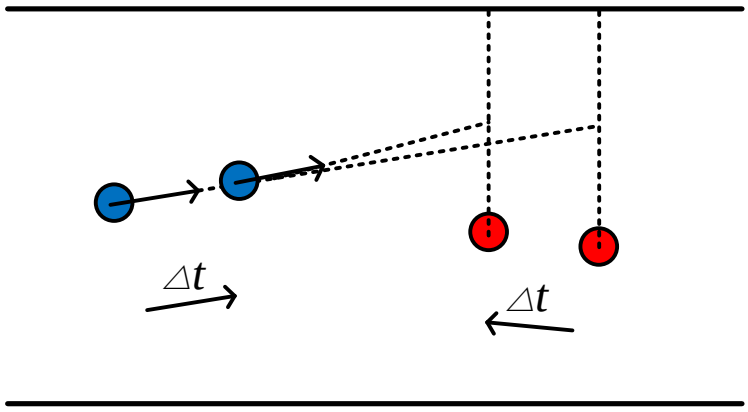

图 6 运动边界等效简化示意图

3.4 假设

假设 1：所有无人机为质点，忽略飞行器大小。
假设 2：红方无人机拦截蓝方无人机后，被拦截蓝方无人机瞬间消失，且红方无人机可继续拦截其他蓝方无人机。

4. 问题建模与求解

4.1 问题一

4.1.1 问题一的建模及算法

对于红方无人机，其初始位置是对称布置，因此为简化建模，只需解出上半区域中的位置即可。根据第三章中的简化，可知当蓝方满足以下约束条件时，蓝方无人机实现成功突防。

$$\begin{cases} \Delta t \le 360s \\ Min(r_1, r_2) > 398m \\ 通过DG1边界或G1G2边界 \end{cases}$$

采用 Matlab 进行程序编程，分别对通过边界 DG1 和 G1G2 的可能性进行判定。当判定是否能突防边界 DG1 时，蓝方无人机采用的突防策略为不断调整速度方向指向实时红方无人机 P1 与运动边界垂线的中点；当判定是否能突防边界 G1G2 时，蓝方无人机采用的突防策略为不断调整速度指向两个红方无人机集群 P1 和 P2 实时位置连线的中点。而红方无人机集群的最优策略为不断调整速度方向垂直于蓝方无人机的实时速度方向。如图 7 所示。红方及蓝方无人机均受到最小转角半径的限制。

将运动区域内所有坐标点代入计算，可算得问题一中蓝方无人机的初始位置区域。

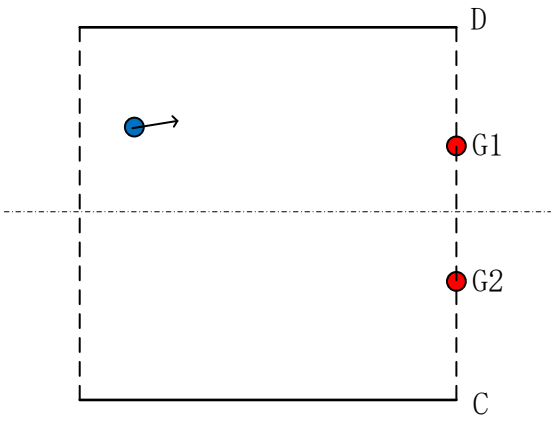

图 7 问题一红蓝双方运动示意图

伪代码如**问题一算法**所示，具体实现 Matlab 代码见附录 6.1，模型代码见附件。

问题一算法

1： ***输入：**目标 E、P1、P2 与边界 PP 坐标

2： ***输出：**满足突防条件的 E 点坐标

3： 指定初始 E、P1、P2 坐标，其中 P1(y)>P2(y)

4： 以 E、P1 为基点绘制阿波罗尼圆，E 与该圆切线的截距 yk1_1>yk1_2；以 E、P2 为基点绘制阿波罗尼圆，E 与该圆切线的截距 yk2_1>yk2_2

5： **if** yk1_2 > yk2_1
6： E 能够突防
7： **else**
8： velocity_E = norm((P1 + P2) / 2 - E) or norm(velocity_E = (P1 + PP) / 2 - E)
9： E = E + velocity_E * dt
10： S1 = P1 - E，S2 = P2 - E
11： velocity_P1 = norm(S1 + velocity_E)， velocity_P1 = norm(S1 + velocity_E)
12： P1= P1 + velocity_P1 * dt, P2= P2 + velocity_P2 * dt
13： **if** dist(P1,E) < dist_min or dist(P2,E) < dist_min
14： E 被拦截
15： **if** E(y) > y_max or T > t_max
16： E 被拦截
17： **if** E(x) > x_max
18： E 能够突防
19： 重复过程 4~18

4.1.2 问题一的运算结果

判定从边界 G1G2 突防可能性的 Matlab 仿真结果图如图 8 所示，两个红方无人机的阿波罗尼圆相切，意味着此时蓝方无人机无突防间隙，必定将被拦截。

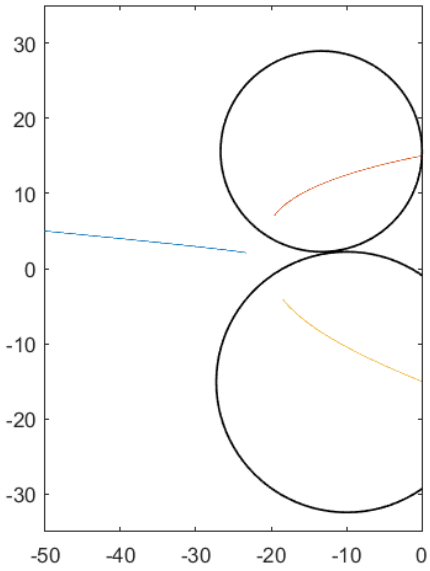

图 8　问题一 G1G2 突防可能性的 Matlab 仿真结果

判定从边界 DG1 突防可能性的 Matlab 运行结果图如图 9 所示，蓝方无人机与红方无人机的阿波罗尼圆切线恰好过 DG1 边界的最上方 D 点，此时蓝方无人机无突防间隙，必定将被拦截。

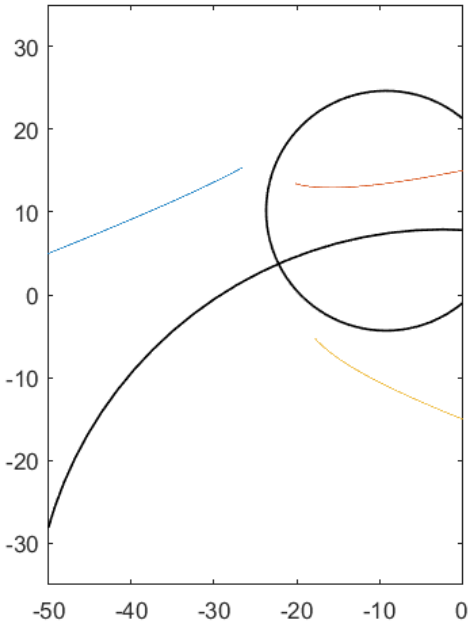

图 9 问题一 DG1 突防可能性的 Matlab 仿真结果

根据限定运动区域的对称性，通过 matlab 运算后，蓝方无人机必定突防区域的上半对称区域如图 10 所示，白色区域为蓝方无人机必定突防的初始区域。黑色区域为不能突防区域，为半径为 $100m$ 的半圆，也即上半区域红方无人机集群的初始圆周阵型范围。由此可见，除红方无人机初始集群圆周范围内，蓝方无人机位于其他任何初始位置时都必定能突防。

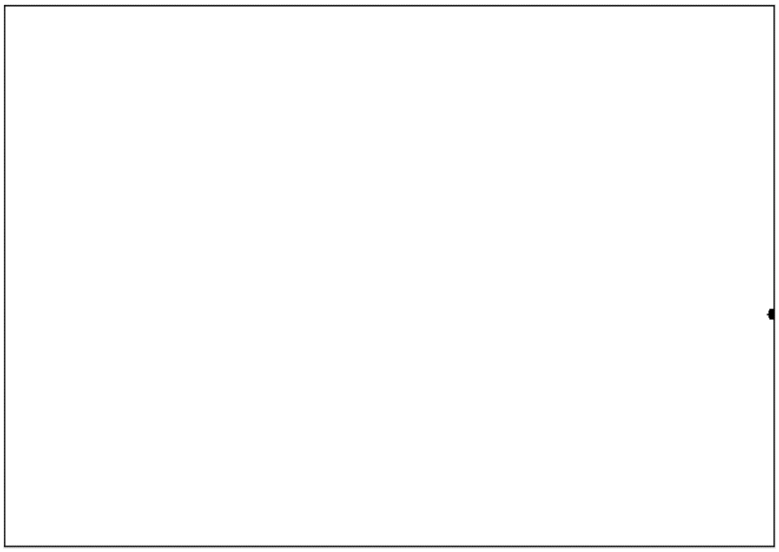

图 10 蓝方无人机必定突防的初始位置

4.2 问题二

4.2.1 问题二的建模及算法

本问题中，蓝方无人机位于 AB 中点，可将红方无人机集群在 CD 上对称布置，Max(L)为最大宽度，则 Max(L_{DG1}) = Max(L_{DG1})，因此只需求出 Max(L_{DG1})与 Max(L_{G1G2})即可。

根据第三章中的简化，可知当蓝方满足以下约束条件时，蓝方无人机实现成功突防。

$$\begin{cases} \Delta t \le 360s \\ Min(r_1, r_2) > 398m \\ 通过DG1边界或G1G2边界 \end{cases}$$

本问题中，由于蓝色无人机以垂直于 CD 的方向直线行驶时，其突防距离最短，因此若 G1G2 边界可突防，则其直线运动方向一定为最优突防策略，因此可以先令 G1G2 之间的距离足够短，红方无人机集群的最优策略为不断调整速度方向垂直于蓝方无人机的实时速度方向，刚好使得蓝色无人机无法直线突防，求出 Max(L_{G1G2})。确定 G1G2 相对于对称轴的位置后，令蓝色无人机改变突防策略为实时速度方向指向红方无人机 G1 实时位置与运动边界垂线的中点。红色无人机集群的最优策略为不断调整速度方向垂直于蓝方无人机的实时速度方向，刚好使得蓝色无人机无法直线突防，求出 Max(L_{DG1})。如图 11 所示。红方及蓝方无人机均受到最小转角半径的限制。

则问题二中的参数

M_{min} = 2Max(L_{DG1}) + Max(L_{G1G2})。

实际通道带宽 M 比 M_{min} 大时，蓝方无人机一定能突破红方无人机集群的拦截。此种情形下蓝方无人机时间最短的突防策略为：（1）当 L_{G1G2} > Max(L_{G1G2})时，以直线方向运动从 G1G2 边界突防。（2）当 L_{G1G2} ≤ Max(L_{G1G2})时，突防策略为速度指向红方无人机 G1 或 G2 的实时位置与运动边界垂线的中点。

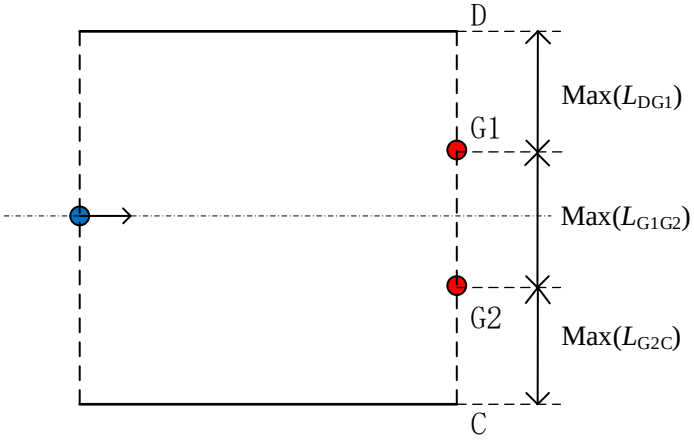

图 11 问题二红蓝双方运动示意图

伪代码如**问题二算法**所示，具体实现 Matlab 代码见附录 6.2，模型代码见附件。

问题二算法
1： ***输入：**目标 E、P1、P2 与边界 PP 坐标
2： ***输出：**满足突防条件的最小 P1、P2 间距

3： 指定初始 E、P1、P2 坐标，其中 P1(y)>P2(y)
4： 以 P1、P2 中点为突围目标，此时 velocity_E = norm((P1 + P2) / 2 - E) = (0.25,0)
5： E = E + velocity_E * dt
6： S1 = P1 - E，S2 = P2 - E
7：velocity_P1 = norm(S1 + velocity_E)，velocity_P2 = norm(S2 + velocity_E)
8： P1= P1 + velocity_P1 * dt, P2= P2 + velocity_P2 * dt
9： **if** dist(P1,E) < dist_min or dist(P2,E) < dist_min
10： E 被拦截
11： **if** E(y) > y_max or T > t_max
12： E 被拦截
13： **if** E(x) > x_max
14： E 能够突防
15： 重复步骤 3~14，得到满足拦截条件的最小 P1、P2 间距
16： ***输入：**目标 E、P1、P2 与边界 PP 坐标
17： ***输出：**满足突防条件的最小边界坐标
18： 以 P1、PP 为突围目标，此时 velocity_E = norm((P1 + PP) / 2 - E)
19： E = E + velocity_E * dt
20： 以 E、P1 为基点绘制阿波罗尼圆，E 与该圆切线的截距 yk1_1>yk1_2
21： **if** yk1_1 < PP(y)
22： E 能够突防
23： S1 = P1 - E，S2 = P2 - E
24： velocity_P1 = norm(S1 + velocity_E)， velocity_P2 = norm(S2 + velocity_E)
25： P1= P1 + velocity_P1 * dt, P2= P2 + velocity_P2 * dt
26： 重复步骤 16~23，得到满足拦截条件的最小 P1、PP 间距

4.2.2 问题二的运算结果

问题二判定从边界 G1G2 突防可能性的 Matlab 仿真结果图如图 12 所示，两个红方无人机的阿波罗尼圆相切，意味着此时蓝方无人机无突防间隙，必定将被拦截。

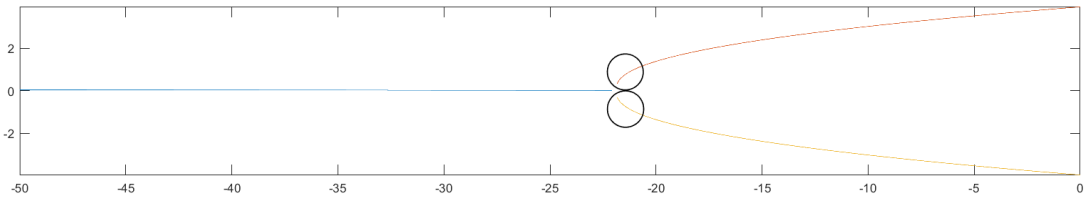

图 12 问题二 G1G2 突防可能性的 Matlab 仿真结果

问题二判定从边界 DG1 突防可能性的 Matlab 运行结果图如图 13 所示，蓝方无人机与红方无人机的阿波罗尼圆切线恰好过 DG1 边界的最上方 D 点，此时蓝方无人机无突防间隙，必定将被拦截。

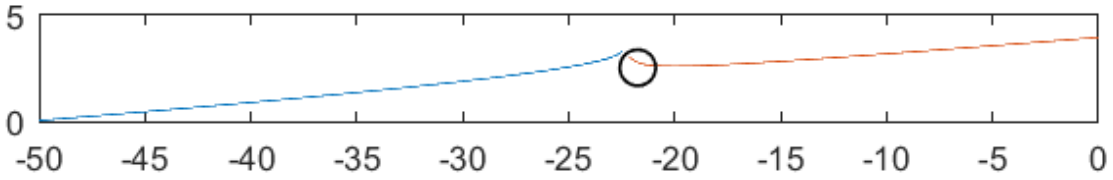

图 13 问题二 DG1 突防可能性的 Matlab 仿真结果

解得 $M_{\min} = 2\text{Max}(L_{DG1}) + \text{Max}(L_{G1G2}) = 0.27 \times 2 + 7.9 = 8.44km$。

4.3 问题三

4.3.1 问题三的建模及算法

本问题中，蓝方无人机位于 AB 中点，可将红方无人机运输机在 CD 上对称布置，Max(L)为最大宽度，因此只需求出 Max(L_{P1D})、Max(L_{G1P1})和 Max(L_{G1G2})即可。

根据第三章中的简化，可知当蓝方满足以下约束条件时，蓝方无人机实现成功突防。

$$\begin{cases} \Delta t \le 360s \\ Min(r_1, r_2) > 398m \\ \text{通过}DP1\text{边界或}P1G1\text{边界或}G1G2\text{边界} \end{cases}$$

本题假设两架运载机两个波次发射的无人机数量均为 5 架，且分布阵型保持半径为 $100m$ 的圆周分布不变，运载机发射第一波无人机集群后，实时速度方向与蓝方突防无人机的实时速度方向保持垂直，这样使得“撒网”面与蓝方无人机的运动方向的夹角最大化，即使拦截成功率最大化，进而使得带宽的上限 $M_{\max}$ 最大化。

本问题可分为三个求解步骤：

（1）根据问题二中的解答，首先确定 $L_{G1G2} = \text{Max}(L_{G1G2})$，即在初始时刻，红方无人机运输机分别在 G1 和 G2 点发射无人机集群，与此同时，蓝方无人机的突防策略为实时速度指向红方无人机 G1 实时位置与运动边界垂线的中点，红方运输机不断调整速度方向使其垂直于蓝方无人机的实时速度方向。如图 14 所示。

（2）假设经过时间 Δt 后，G1 点到达 G1’点，此时红方运输机从 G1 点到达 P1 点，并在 P1 点发射第二波无人机集群，第二波无人机集群的初始速度方向与蓝方无人机实时速度方向垂直。此时蓝方无人机改变突防策略为实时调整速度指向两个红方无人机集群 P1 点和 G1’点实时位置连线的中点。求出 Δt，使得蓝方无人机刚好无法从边界 P1G1 突防。求出 Δt 后，可由 P1 的坐标求出 Max(L_{G1P1})。如图 15 所示。

（3）由（2）中得到的时间 Δt 来布置红方第一波及第二波无人机集群，G1 点到达 G1’点，此时红方运输机从 G1 点到达 P1 点，并在 P1 点发射第二波无人机集群，第二波无人机集群的初始速度方向与蓝方无人机实时速度方向垂直。此时蓝方无人机改变突防策略为实时速度方向指向红方无人机 G1 实时位置与运动边界垂线的中点。使得蓝方无人机刚好无法从边界 P1G1 突防，求出 Max(L_{P1D})。如图 16 所示。红方及蓝方无人机均受到最小转角半径的限制。

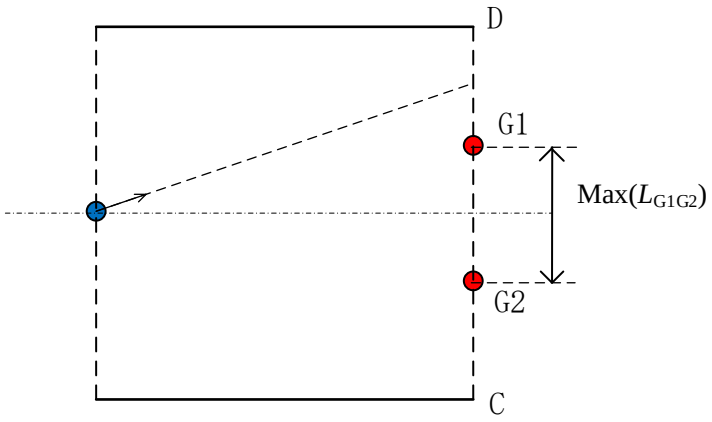

图 14 问题三红蓝双方运动示意图一

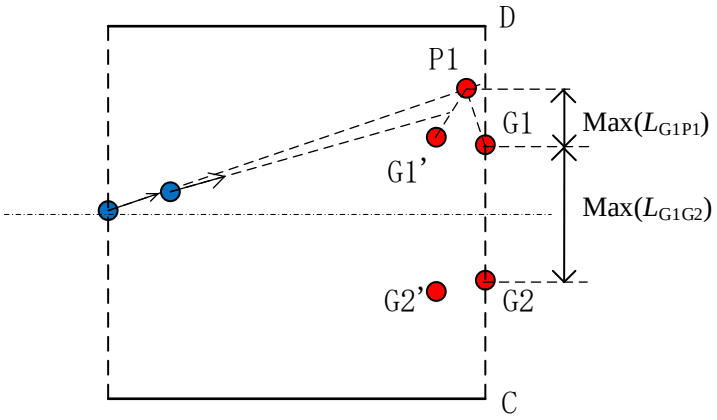

图 15 问题三红蓝双方运动示意图二

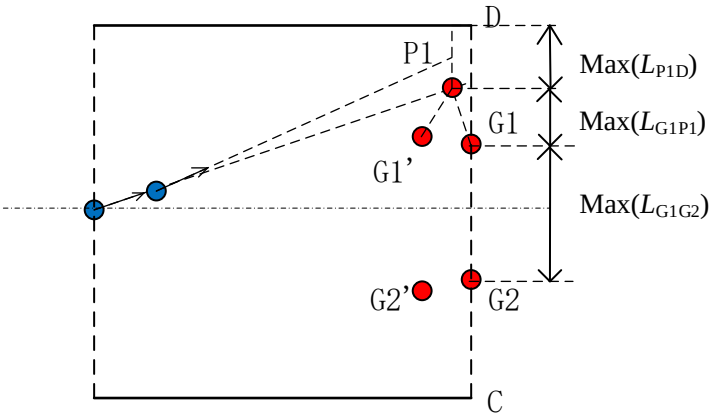

图 16 问题三红蓝双方运动示意图三

则问题三中的参数

$M_{max} = 2(\text{Max}(L_{G1P1}) + \text{Max}(L_{P1D})) + \text{Max}(L_{G1G2})$

伪代码如**题目三算法**所示，具体实现 Matlab 代码见附录 6.3，模型代码见附件。

题目三算法

1： ***输入：**目标 E、P1、P2 与边界 PP 坐标
2： ***输出：**满足突防条件的最小边界坐标
3： 指定初始 E、P1、P2 坐标，其中 P1(y)>P2(y)
4： 以 P1、P2 中点为突围目标，此时 P1_P2_min = 7.9 km
5： 设 E 初始向上突围，运输机 G1 速度方向保持与 E 垂直，在 t 时刻 G1 投放 P3
6： E = E + velocity_E * dt
7： S1 = P1 - E，S3 = P3 - E
8： velocity_P1 = norm(S1 + velocity_E)， velocity_P3 = norm(S3 + velocity_E)，
9： P1= P1 + velocity_P1 * dt，P3= P3 + velocity_P3 * dt
10： **if** dist(P1,E) < dist_min or dist(P3,E) < dist_min
11： E 被捕获
12： 重复步骤 5~11，得到满足拦截条件的时刻 t
13： ***输入：**t 时刻目标 E、P3 与边界 PP 坐标
14： ***输出：**满足突防条件的最小边界坐标
15： 以 P3、PP 为突围目标，此时 velocity_E = norm((P3 + PP) / 2 - E)
16： E = E + velocity_E * dt
17： 以 E、P3 为基点绘制阿波罗尼圆，E 与该圆切线的截距 yk1_1>yk1_2
18： **if** yk1_1 < PP(y)
19： E 能够突防
20： S3 = P3 - E
21： velocity_P3 = norm(S3 + velocity_E)
22： P3= P3 + velocity_P3 * dt
23： 重复步骤 15~22，得到满足拦截条件的最小 PP 坐标

4.3.2 问题三的运算结果

问题三 G1P1 突防可能性的 Matlab 仿真结果如图 17 所示。

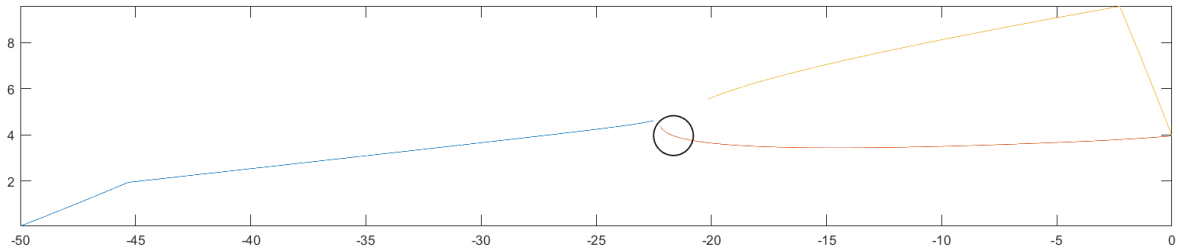

图 17 问题三 G1P1 突防可能性的 Matlab 仿真结果

问题三 DP1 突防可能性的 Matlab 仿真结果如图 18 所示。由图中可以看出第一波红色

无人机集群出现了回转拦截现象，由此说明红色无人机集群的多波次协同拦截将大大增加拦截成功率。

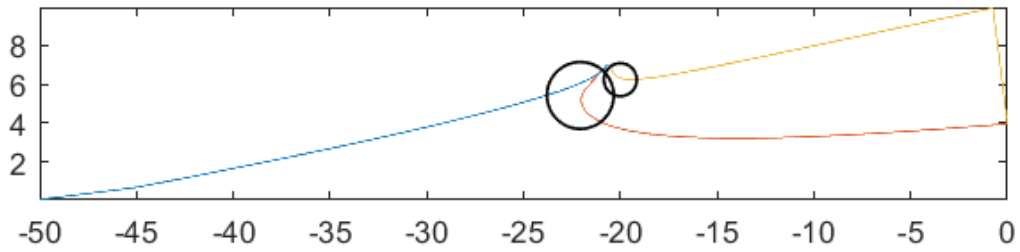

图 18 问题三 DP1 突防可能性的 Matlab 仿真结果

解得 $M_{\max} = 2(\mathrm{Max}(L_{G1P1}) + \mathrm{Max}(L_{P1D})) + \mathrm{Max}(L_{G1G2}) = 2\times（0.66 + 3.65）+ 7.9 = 16.52km$。

5. 参考文献

[1] 严富函. 考虑追捕者能力不确定性的多 Agent 追逃问题研究[D]. 2019.
[2] 王泉德. 机器学习及其在多 Agent 对策学习中的应用研究[D].武汉大学,2005.
[3] 陈侠, 李光耀, 于兴超. 基于博弈策略的多无人机航迹规划研究[J]. 战术导弹技术, 2020(1).
[4] 李登峰. 微分对策及其应用[M]. 国防工业出版社, 180-190.2000.
[5] 查文中. 单个优势逃跑者的多人定性微分对策研究[D].北京理工大学,2016.
[6] Pachter M, Garcia E, Casbeer D W. Active target defense differential game[C]. Communication, Control, & Computing. IEEE, 2015.
[7] Yamasaki T, Balakrishnan S. Triangle Intercept Guidance for Aerial Defense[J]. 2010.
[8] Rusnak I, Weiss H, Hexner G. Guidance laws in target-Missile-Defender scenario with an aggressive defender[J]. IFAC Proceedings Volumes, 2011, 44(1): 9349-9354.

6. 附录

6.1 问题一代码

```
clc;
clear;
close all;
%%
%初始条件
count_num = 10000;
x0 = 0;
y0 = 15;
x1 = -50;
y1 = 10;
mat = zeros(350,500);%存活点
parfor ii = 1:500
    x1 = -50 + 0.1 * ii;
    for jj = 1:350
        y1 = 0.1 * jj;
        x00 = 0;
        y00 = 35;
        x01 = 0;
        y01 = 0;
        dt = 0.1;
        J = [1,0];
        disp_min = sqrt((x0 - x1)^2 + (y0 - y1)^2);
        %%
        P = [x0,y0];
        P1 = [x0,-y0];
        E = [x1,y1];
        PP = [x00,y00];
        S_ini = [(x0 + PP(1)) / 2 - x1,(y0 + PP(2)) / 2 - y1];
        ve = 0.25 * norm_1(S_ini);
        E = E + ve * dt;            %E 点坐标
        S = E - P;                  %两点距离
        S1 = E - P1;
        vp = 0.2 * norm_1(S + ve * dt);%p 速度
        vp1 = 0.2 * norm_1(S1 + ve * dt);
        P = P + vp * dt;            %P 点坐标
        P1 = P1 + vp * dt;
        for i = 1:count_num
%                   ex(i) = E(1);
```

```
%                   ey(i) = E(2);
%                   px(i) = P(1);
%                   py(i) = P(2);
%                   px1(i) = P1(1);
%                   py1(i) = P1(2);
                disp = sqrt(S(1)^2 + S(2)^2);
                disp1 = sqrt(S1(1)^2 + S1(2)^2);
                logc = escape(E,P,PP);
                if disp < 0.4 || disp1 < 0.4
                    break
                end
                if i >= 3600
                    break
                end
                if E(2) >= 35
                    break
                elseif E(1)>= 0
                    mat(jj,ii) = 100;
                    break
                end
                if logc ~= [0,0]
                    ve_temp = 0.25 * norm_1(logc);%e 速度
                else
                    t_rem = 360 - dt * i;
                    t_least = abs(E(1) / ve(1));
                    if t_least < t_rem
                        mat(jj,ii) = 100;
                    end
                    break
                end
                theta_1 = get_direc(E,PP,ve_temp) * acos((ve * ve_temp') / (sqrt(ve(1)^2 + ve(2)^2) *
sqrt(ve_temp(1)^2 + ve_temp(2)^2)));            %E 偏转角
                if abs(theta_1) > asin(dt/4)
                    theta_1 = asin(dt/4) * theta_1 / abs(theta_1);
                end
                ve = rotate_1(ve,theta_1);          %E 点坐标
                E = E + ve * dt;
                S = E - P;                  %两点距离
                S1 = E - P1;
                vp_temp = 0.2 * norm_1(S + ve);     %p 速度
                vp_temp1 = 0.2 * norm_1(S1 + ve);
```

```
                theta_2 = get_direc(P,E,vp_temp) * acos((vp * vp_temp') / (sqrt(vp(1)^2 + vp(2)^2) *
sqrt(vp_temp(1)^2 + vp_temp(2)^2)));          %P 偏转角
                theta_21 = get_direc(P1,E,vp_temp1) * acos((vp1 * vp_temp1') / (sqrt(vp1(1)^2 + vp1(2)^2) *
sqrt(vp_temp1(1)^2 + vp_temp1(2)^2)));
                if abs(theta_2) > asin(2*dt/7)
                    theta_2 = asin(dt/5) * theta_2 / abs(theta_2);
                end
                if abs(theta_21) > asin(2*dt/7)
                    theta_21 = asin(dt/5) * theta_21 / abs(theta_21);
                end
                vp = rotate_1(vp,theta_2);          %P 点坐标
                vp1 = rotate_1(vp1,theta_2);
                P = P + vp * dt;
                P1 = P1 + vp1 * dt;
            end
            %%
            P = [x0,y0];
            P1 = [x0,-y0];
            E = [x1,y1];
            PP = [x01,y01];
            S_ini = [(x0 + P1(1)) / 2 - x1,(y0 + P1(2)) / 2 - y1];
            ve = 0.25 * norm_1(S_ini);
            E = E + ve * dt;          %E 点坐标
            S = E - P;                %两点距离
            S1 = E - P1;
            vp = 0.2 * norm_1(S + ve * dt);%p 速度
            vp1 = 0.2 * norm_1(S1 + ve * dt);
            P = P + vp * dt;          %P 点坐标
            P1 = P1 + vp * dt;
            for i = 1:count_num
    %           ex(i) = E(1);
    %           ey(i) = E(2);
    %           px(i) = P(1);
    %           py(i) = P(2);
    %           px1(i) = P1(1);
    %           py1(i) = P1(2);
                disp = sqrt(S(1)^2 + S(2)^2);
                disp1 = sqrt(S1(1)^2 + S1(2)^2);
                logc = escape_1(E,P,P1);
                if disp < 0.4 || disp1 < 0.4
                    break
                end
```

```
                if i >= 3600
                    break
                end
                if E(2) >= 35
                    break
                elseif E(1)>= 0
                    mat(jj,ii) = 100;
                    break
                end
                if logc ~= [0,0]
                    ve_temp = 0.25 * norm_1(logc);%e 速度
                else
                    mat(jj,ii) = 100;
                    break
                end
                theta_1 = acos((ve * ve_temp') / (sqrt(ve(1)^2 + ve(2)^2) * sqrt(ve_temp(1)^2 +
ve_temp(2)^2)));          %E 偏转角
                if abs(theta_1) > asin(dt/4)
                    theta_1 = asin(dt/4) * theta_1 / abs(theta_1);
                end
                ve = rotate_1(ve,theta_1);              %E 点坐标
                E = E + ve * dt;
                S = E - P;                      %两点距离
                S1 = E - P1;
                vp_temp = 0.2 * norm_1(S + ve);     %p 速度
                vp_temp1 = 0.2 * norm_1(S1 + ve);
                theta_2 = get_direc(P,E,vp_temp) * acos((vp * vp_temp') / (sqrt(vp(1)^2 + vp(2)^2) *
sqrt(vp_temp(1)^2 + vp_temp(2)^2)));              %P 偏转角
                theta_21 = get_direc(P1,E,vp_temp1) * acos((vp1 * vp_temp1') / (sqrt(vp1(1)^2 + vp1(2)^2) *
sqrt(vp_temp1(1)^2 + vp_temp1(2)^2)));
                if abs(theta_2) > asin(dt/5)
                    theta_2 = asin(dt/5) * theta_2 / abs(theta_2);
                end
                if abs(theta_21) > asin(dt/5)
                    theta_21 = asin(dt/5) * theta_21 / abs(theta_21);
                end
                vp = rotate_1(vp,theta_2);              %P 点坐标
                vp1 = rotate_1(vp1,theta_21);
                P = P + vp * dt;
                P1 = P1 + vp1 * dt;
            end
        end
```

```
end
%%
% plot(ex,ey,px,py);
% axis equal
%%
imshow(mat);
function y = norm_1(x)
a = x(1);
b = x(2);
val = sqrt(a^2 + b^2);
y = x / val;
end
function y = get_direc(e,p,n)
ya = n(2) / n(1) * (p(1) - e(1)) + e(2) - p(2);
if ya ~= 0
    y = ya / abs(ya);
else
    y = ya;
end
end
function y = rotate_1(a,n)
T = [cos(n),sin(n);-sin(n),cos(n)];
A = T * a';
y = [A(1),A(2)];
end
function y = escape(e,p,pp)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
point = pp;
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
```

```
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 < 35
    y = [0,0];%逃离
else
    y = [x0 - x1,(y0 + point(2))/2 - y1];%沿中点运动
end
end
function y = escape_1(e,p,p1)
x0 = p(1);
y0 = p(2);
x01 = p1(1);
y01 = p1(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y4 - (y4 - y1)/(x4 - x1) * x4;%切线与 y 轴交点
```

```
%%
dist = sqrt((x1-x01)^2+(y1-y01)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x01 - x1);%圆心坐标
y2 = y1 + h * (y01 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk2 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 > yk2
    y = [0,0];%逃离
else
    y = [(x0 + x01)/2 - x1,(y0 + y01)/2 - y1];%沿中点运动
end
end
```

6.1.1 问题二代码第一段

```
clc;
clear;
close all;
%%
%初始条件
count_num = 10000;
x0 = 0;
for iii = 1:500
    y0 = 0.01 * iii;
    x1 = -50;
    y1 = 0.05;
    x00 = 0;
    y00 = 35;
    x01 = 0;
    y01 = 0;
    dt = 0.1;
    J = [1,0];
    disp_min = sqrt((x0 - x1)^2 + (y0 - y1)^2);
    %%
    P = [x0,y0];
    P1 = [x0,-y0];
    E = [x1,y1];
    PP = [x01,y01];
    S_ini = [(x0 + P1(1)) / 2 - x1,(y0 + P1(2)) / 2 - y1];
    ve = 0.25 * norm_1(S_ini);
    E = E + ve * dt;            %E 点坐标
    S = E - P;                  %两点距离
    S1 = E - P1;
    vp = 0.2 * norm_1(S + ve * dt);%p 速度
    vp1 = 0.2 * norm_1(S1 + ve * dt);
    P = P + vp * dt;            %P 点坐标
    P1 = P1 + vp * dt;
    for i = 1:count_num
        ex(i) = E(1);
        ey(i) = E(2);
        px(i) = P(1);
        py(i) = P(2);
        px1(i) = P1(1);
        py1(i) = P1(2);
        disp = sqrt(S(1)^2 + S(2)^2);
```

```
            disp1 = sqrt(S1(1)^2 + S1(2)^2);
            logc = escape_1(E,P,P1);
            if disp < 0.4 || disp1 < 0.4
                get = 1;
                break
            end
            if i >= 3600
                get = 1;
                break
            end
            if E(2) >= y00
                get = 1;
                break
            elseif E(1)>= 0
                get = 0;
                break
            end
            if logc ~= [0,0]
                ve_temp = 0.25 * norm_1(logc);%e 速度
            else
                t_rem = 360 - dt * i;
                t_least = abs(E(1) / ve(1));
                if t_least < t_rem
                    get = 0;
                else
                    get = 1;
                end
                break
            end
            ve = ve_temp;              %E 点坐标
            E = E + ve * dt;
            S = E - P;                      %两点距离
            S1 = E - P1;
            vp_temp = 0.2 * norm_1(S + ve);     %p 速度
            vp_temp1 = 0.2 * norm_1(S1 + ve);
            theta_2 = get_direc(P,E,vp_temp) * acos((vp * vp_temp') / (sqrt(vp(1)^2 + vp(2)^2) *
sqrt(vp_temp(1)^2 + vp_temp(2)^2)));              %P 偏转角
            theta_21 = get_direc(P1,E,vp_temp1) * acos((vp1 * vp_temp1') / (sqrt(vp1(1)^2 + vp1(2)^2) *
sqrt(vp_temp1(1)^2 + vp_temp1(2)^2)));
            if abs(theta_2) > asin(dt/5)
                theta_2 = asin(dt/5) * theta_2 / abs(theta_2);
            end
```

```
            if abs(theta_21) > asin(dt/5)
                theta_21 = asin(dt/5) * theta_21 / abs(theta_21);
            end
            vp = rotate_1(vp,theta_2);              %P 点坐标
            vp1 = rotate_1(vp1,theta_21);
            P = P + vp * dt;
            P1 = P1 + vp1 * dt;
        end
        get1(iii) = get;
end
%%
% plot(ex,ey,px,py,px1,py1);
% axis equal
% circle(E,P);
% circle(E,P1);
% axis([-50,0,-inf,inf]);
%%
function [] = circle(e,p)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
rectangle('Position',[x2-r,y2-r,2*r,2*r],'Curvature',[1,1],'linewidth',1),axis equal
end
function y = norm_1(x)
a = x(1);
b = x(2);
val = sqrt(a^2 + b^2);
y = x / val;
end
function y = get_direc(e,p,n)
ya = n(2) / n(1) * (p(1) - e(1)) + e(2) - p(2);
if ya ~= 0
    y = ya / abs(ya);
else
    y = ya;
```

```
end
end
function y = rotate_1(a,n)
T = [cos(n),sin(n);-sin(n),cos(n)];
A = T * a';
y = [A(1),A(2)];
end
function y = escape(e,p,pp)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
point = pp;
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 < 35
    y = [0,0];%逃离
else
    y = [x0 - x1,(y0 + point(2))/2 - y1];%沿中点运动
end
end
function y = escape_1(e,p,p1)
x0 = p(1);
y0 = p(2);
```

```
x01 = p1(1);
y01 = p1(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y4 - (y4 - y1)/(x4 - x1) * x4;%切线与 y 轴交点
%%
dist = sqrt((x1-x01)^2+(y1-y01)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x01 - x1);%圆心坐标
y2 = y1 + h * (y01 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
```

```
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk2 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 > yk2
    y = [0,0];%逃离
else
    y = [(x0 + x01)/2 - x1,(y0 + y01)/2 - y1];%沿中点运动
end
end
```

6.2.2 问题二代码第二段

```
clc;
clear;
close all;
%%
%初始条件
count_num = 10000;
x0 = 0;
y0 = 3.95;
x1 = -50;
y1 = 0.05;
x00 = 0;
for iii = 1:100
y00 = 3.95 + iii * 0.01;
x01 = 0;
y01 = 0;
dt = 0.1;
J = [1,0];
disp_min = sqrt((x0 - x1)^2 + (y0 - y1)^2);
%%
P = [x0,y0];
P1 = [x0,-y0];
E = [x1,y1];
PP = [x00,y00];
S_ini = [(x00 + x0) / 2 - x1,(y00 + y0) / 2 - y1];
ve = 0.25 * norm_1(S_ini);
E = E + ve * dt;              %E 点坐标
S = E - P;                    %两点距离
vp = 0.2 * norm_1(S + ve * dt);%p 速度
P = P + vp * dt;              %P 点坐标
for i = 1:count_num
    ex(i) = E(1);
    ey(i) = E(2);
    px(i) = P(1);
    py(i) = P(2);
    disp = sqrt(S(1)^2 + S(2)^2);
    logc = escape(E,P,PP);
    if disp < 0.4
        get = 1;
        break
    end
```

```
        if i >= 3600
            get = 1;
            break
        end
        if E(2) >= y00
            get = 1;
            break
        elseif E(1)>= 0
            get = 0;
            break
        end
        if logc ~= [0,0]
            ve_temp = 0.25 * norm_1(logc);%e 速度
        else
            t_rem = 360 - dt * i;
            t_least = abs(E(1) / ve(1));
            if t_least < t_rem
                get = 0;
            else
                get = 1;
            end
            break
        end
        ve = ve_temp;              %E 点坐标
        E = E + ve * dt;
        S = E - P;                    %两点距离
        vp_temp = 0.2 * norm_1(S + ve);     %p 速度
        theta_2 = get_direc(P,E,vp_temp) * acos((vp * vp_temp') / (sqrt(vp(1)^2 + vp(2)^2) * sqrt(vp_temp(1)^2
+ vp_temp(2)^2)));             %P 偏转角
        if abs(theta_2) > asin(dt/5)
            theta_2 = asin(dt/5) * theta_2 / abs(theta_2);
        end
        vp = rotate_1(vp,theta_2);              %P 点坐标
        P = P + vp * dt;
    end
    get1(iii) = get;
    end
    %%
    % plot(ex,ey,px,py);
    % axis equal
    % circle(E,P);
    %%
```

```
function [] = circle(e,p)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
rectangle('Position',[x2-r,y2-r,2*r,2*r],'Curvature',[1,1],'linewidth',1),axis equal
end
function y = norm_1(x)
a = x(1);
b = x(2);
val = sqrt(a^2 + b^2);
y = x / val;
end
function y = get_direc(e,p,n)
ya = n(2) / n(1) * (p(1) - e(1)) + e(2) - p(2);
if ya ~= 0
    y = ya / abs(ya);
else
    y = ya;
end
end
function y = rotate_1(a,n)
T = [cos(n),sin(n);-sin(n),cos(n)];
A = T * a';
y = [A(1),A(2)];
end
function y = escape(e,p,pp)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
point = pp;
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
```

```
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 < point(2)
    y = [0,0];%逃离
else
    y = [x0 - x1,(y0 + point(2))/2 - y1];%沿中点运动
end
end
function y = escape_1(e,p,p1)
x0 = p(1);
y0 = p(2);
x01 = p1(1);
y01 = p1(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
```

```
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y4 - (y4 - y1)/(x4 - x1) * x4;%切线与 y 轴交点
%%
dist = sqrt((x1-x01)^2+(y1-y01)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x01 - x1);%圆心坐标
y2 = y1 + h * (y01 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk2 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 > yk2
    y = [0,0];%逃离
else
    y = [(x0 + x01)/2 - x1,(y0 + y01)/2 - y1];%沿中点运动
end
end
```

6.3.1 问题三代码第一段

```
clc;
clear;
close all;
%%
%初始条件
count_num = 10000;
x0 = 0;
y0 = 3.95;
x1 = -50;
y1 = 0.05;
x00 = 0;
y00 = 35;
x01 = 0;
y01 = 0;
dt = 0.1;
J = [1,0];
disp_min = sqrt((x0 - x1)^2 + (y0 - y1)^2);
%%
for iii = 1:300
P = [x0,y0];
P1 = [x0,y0];
E = [x1,y1];
PP = [x00,y00];
PP1 = [x01,y01];
S_ini = [(x00 + x0) / 2 - x1,(y00 + y0) / 2 - y1];
ve = 0.25 * norm_1(S_ini);
E = E + ve * dt;              %E 点坐标
S = E - P;                    %两点距离
vp = 0.2 * norm_1(S + ve * dt);%p 速度
S_ini1 = [-ve(2),ve(1)];
vp1 = 0.3 * norm_1(S_ini1);
P = P + vp * dt;              %P 点坐标
P1 = P1 + vp1 * dt;

    for i = 1:iii
        ex(i) = E(1);
        ey(i) = E(2);
        px(i) = P(1);
        py(i) = P(2);
        px1(i) = P1(1);
```

```
py1(i) = P1(2);
disp = sqrt(S(1)^2 + S(2)^2);
logc = escape(E,P,PP);
if disp < 0.4
    get = 1;
    break
end
if i >= 3600
    get = 1;
    break
end
if E(2) >= y00
    get = 1;
    break
elseif E(1)>= 0
    get = 0;
    break
end
if logc ~= [0,0]
    ve_temp = 0.25 * norm_1(logc);%e 速度
else
    t_rem = 360 - dt * i;
    t_least = abs(E(1) / ve(1));
    if t_least < t_rem
        get = 0;
    else
        get = 1;
    end
    break
end
theta_1 = get_direc(E,PP,ve_temp) * acos((ve * ve_temp') / (sqrt(ve(1)^2 + ve(2)^2) *
sqrt(ve_temp(1)^2 + ve_temp(2)^2)));          %E 偏转角
if abs(theta_1) > asin(dt/4)
    theta_1 = asin(dt/4) * theta_1 / abs(theta_1);
end
ve = rotate_1(ve,theta_1);          %E 点坐标
E = E + ve * dt;
S = E - P;                    %两点距离
vp_temp = 0.2 * norm_1(S + ve);    %p 速度
S_ini1 = [-ve(2),ve(1)];
vp1 = 0.3 * norm_1(S_ini1);
```

```
            theta_2 = get_direc(P,E,vp_temp) * acos((vp * vp_temp') / (sqrt(vp(1)^2 + vp(2)^2) *
sqrt(vp_temp(1)^2 + vp_temp(2)^2)));              %P 偏转角
            if abs(theta_2) > asin(dt/5)
                theta_2 = asin(dt/5) * theta_2 / abs(theta_2);
            end
            vp = rotate_1(vp,theta_2);              %P 点坐标
            P = P + vp * dt;
            P1 = P1 + vp1 * dt;
        end
        S = E - P;                      %两点距离
        S1 = E - P1;
        for j = 1:count_num
            ex(i+j) = E(1);
            ey(i+j) = E(2);
            px(i+j) = P(1);
            py(i+j) = P(2);
            px1(i+j) = P1(1);
            py1(i+j) = P1(2);
            disp = sqrt(S(1)^2 + S(2)^2);
            disp1 = sqrt(S1(1)^2 + S1(2)^2);
            logc = escape_1(E,P,P1);
            if disp < 0.4 || disp1 < 0.4
                get = 1;
                break
            end
            if i >= 3600 - iii
                get = 2;
                break
            end
            if E(2) >= y00
                get = 3;
                break
            elseif E(1)>= 0
                get = 0;
                break
            end
            if logc ~= [0,0]
                ve_temp = 0.25 * norm_1(logc);%e 速度
            else
                t_rem = 360 - dt * (i + j);
                t_least = abs(E(1) / ve(1));
                if t_least < t_rem
```

```
                get = 0;
            else
                get = 4;
            end
            break
        end
        ve = ve_temp;              %E 点坐标
        E = E + ve * dt;
        S = E - P;                 %两点距离
        S1 = E - P1;
        vp = 0.2 * norm_1(S + ve);     %p 速度
        vp1 = 0.2 * norm_1(S1 + ve);
        P = P + vp * dt;
        P1 = P1 + vp1 * dt;
    end
    get_1(iii) = get;
end
%%
% plot(ex,ey,px,py,px1,py1);
% axis equal
% circle(E,P);
% axis([-50,0,-inf,inf]);
%%
function [] = circle(e,p)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
rectangle('Position',[x2-r,y2-r,2*r,2*r],'Curvature',[1,1],'linewidth',1),axis equal
end
function y = norm_1(x)
a = x(1);
b = x(2);
val = sqrt(a^2 + b^2);
y = x / val;
end
```

```
function y = get_direc(e,p,n)
ya = n(2) / n(1) * (p(1) - e(1)) + e(2) - p(2);
if ya ~= 0
    y = ya / abs(ya);
else
    y = ya;
end
end
function y = rotate_1(a,n)
T = [cos(n),sin(n);-sin(n),cos(n)];
A = T * a';
y = [A(1),A(2)];
end
function y = escape(e,p,pp)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
point = pp;
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 < point(2)
    y = [0,0];%逃离
else
```

```
    y = [x0 - x1,(y0 + point(2))/2 - y1];%沿中点运动
end
end
function y = escape_1(e,p,p1)
x0 = p(1);
y0 = p(2);
x01 = p1(1);
y01 = p1(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y4 - (y4 - y1)/(x4 - x1) * x4;%切线与 y 轴交点
%%
dist = sqrt((x1-x01)^2+(y1-y01)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x01 - x1);%圆心坐标
y2 = y1 + h * (y01 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
```

```
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk2 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 > yk2
    y = [0,0];%逃离
else
    y = [(x0 + x01)/2 - x1,(y0 + y01)/2 - y1];%沿中点运动
end
end
```

6.3.2 问题三代码第二段

```
clc;
clear;
close all;
%%
%初始条件
for iii = 1:1000
    ex = zeros(1);
    ey = zeros(1);
    px = zeros(1);
    py = zeros(1);
    px1 = zeros(1);
    py1 = zeros(1);
    count_num = 10000;
    x0 = 0;
    y0 = 3.95;
    x1 = -50;
    y1 = 0.05;
    x00 = 0;
    y00 = 35;
    x01 = 0;
    y01 = 0;
    dt = 0.1;
    J = [1,0];
    disp_min = sqrt((x0 - x1)^2 + (y0 - y1)^2);
    %%
    y00 = 5 + iii * 0.01;
    P = [x0,y0];
    P1 = [x0,y0];
    E = [x1,y1];
    PP = [x00,y00];
    PP1 = [x01,y01];
    S_ini = [(x00 + x0) / 2 - x1,(y00 + y0) / 2 - y1];
    ve = 0.25 * norm_1(S_ini);
    E = E + ve * dt;              %E 点坐标
    S = E - P;                    %两点距离
    vp = 0.2 * norm_1(S + ve * dt);%p 速度
    S_ini1 = [-ve(2),ve(1)];
    vp1 = 0.3 * norm_1(S_ini1);
    P = P + vp * dt;              %P 点坐标
    P1 = P1 + vp1 * dt;
```

```
for i = 1:200
    ex(i) = E(1);
    ey(i) = E(2);
    px(i) = P(1);
    py(i) = P(2);
    px1(i) = P1(1);
    py1(i) = P1(2);
    disp = sqrt(S(1)^2 + S(2)^2);
    logc = escape(E,P,PP);
    if disp < 0.4
        get = 1;
        break
    end
    if i >= 3600
        get = 1;
        break
    end
    if E(2) >= y00
        get = 1;
        break
    elseif E(1)>= 0
        get = 0;
        break
    end
    if logc ~= [0,0]
        ve_temp = 0.25 * norm_1(logc);%e 速度
    else
        t_rem = 360 - dt * i;
        t_least = abs(E(1) / ve(1));
        if t_least < t_rem
            get = 0;
        else
            get = 1;
        end
        break
    end
    theta_1 = get_direc(E,PP,ve_temp) * acos((ve * ve_temp') / (sqrt(ve(1)^2 + ve(2)^2) *
sqrt(ve_temp(1)^2 + ve_temp(2)^2)));          %E 偏转角
    if abs(theta_1) > asin(dt/4)
        theta_1 = asin(dt/4) * theta_1 / abs(theta_1);
    end
    ve = rotate_1(ve,theta_1);          %E 点坐标
```

```
            E = E + ve * dt;
            S = E - P;                      %两点距离
            vp_temp = 0.2 * norm_1(S + ve);     %p 速度
            S_ini1 = [-ve(2),ve(1)];
            vp1 = 0.3 * norm_1(S_ini1);
            theta_2 = get_direc(P,E,vp_temp) * acos((vp * vp_temp') / (sqrt(vp(1)^2 + vp(2)^2) *
sqrt(vp_temp(1)^2 + vp_temp(2)^2)));            %P 偏转角
            if abs(theta_2) > asin(dt/5)
                theta_2 = asin(dt/5) * theta_2 / abs(theta_2);
            end
            vp = rotate_1(vp,theta_2);          %P 点坐标
            P = P + vp * dt;
            P1 = P1 + vp1 * dt;
        end
        S = E - P;                      %两点距离
        S1 = E - P1;
        for j = 1:count_num
            ex(i+j) = E(1);
            ey(i+j) = E(2);
            px(i+j) = P(1);
            py(i+j) = P(2);
            px1(i+j) = P1(1);
            py1(i+j) = P1(2);
            disp = sqrt(S(1)^2 + S(2)^2);
            disp1 = sqrt(S1(1)^2 + S1(2)^2);
            logc = escape(E,P1,PP);
            if disp < 0.4 || disp1 < 0.4
                get = 1;
                break
            end
            if j >= 3600 - i
                get = 2;
                break
            end
            if E(2) >= y00
                get = 3;
                break
            elseif E(1)>= 0
                get = 0;
                break
            end
            if logc ~= [0,0]
```

```
            ve_temp = 0.25 * norm_1(logc);%e 速度
        else
            t_rem = 360 - dt * (i + j);
            t_least = abs(E(1) / ve(1));
            if t_least < t_rem
                get = 0;
            else
                get = 4;
            end
            break
        end
        ve = ve_temp;              %E 点坐标
        E = E + ve * dt;
        S = E - P;                      %两点距离
        S1 = E - P1;
        vp = 0.2 * norm_1(S + ve);      %p 速度
        vp1 = 0.2 * norm_1(S1 + ve);
        P = P + vp * dt;
        P1 = P1 + vp1 * dt;
    end
    get_1(iii) = get;
end
%%
plot(ex,ey,px,py,px1,py1);
axis equal
circle(E,P);
axis([-50,0,-inf,inf]);
%%
function [] = circle(e,p)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
rectangle('Position',[x2-r,y2-r,2*r,2*r],'Curvature',[1,1],'linewidth',1),axis equal
end
function y = norm_1(x)
```

```
a = x(1);
b = x(2);
val = sqrt(a^2 + b^2);
y = x / val;
end
function y = get_direc(e,p,n)
ya = n(2) / n(1) * (p(1) - e(1)) + e(2) - p(2);
if ya ~= 0
    y = ya / abs(ya);
else
    y = ya;
end
end
function y = rotate_1(a,n)
T = [cos(n),sin(n);-sin(n),cos(n)];
A = T * a';
y = [A(1),A(2)];
end
function y = escape(e,p,pp)
x0 = p(1);
y0 = p(2);
x1 = e(1);
y1 = e(2);
point = pp;
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
```

```
x4 = (c-b*y4)/a;
yk1 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 < point(2)
    y = [0,0];%逃离
else
    y = [x0 - x1,(y0 + point(2))/2 - y1];%沿中点运动
end
end
function y = escape_1(e,p,p1)
x0 = p(1);
y0 = p(2);
x01 = p1(1);
y01 = p1(2);
x1 = e(1);
y1 = e(2);
dist = sqrt((x1-x0)^2+(y1-y0)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x0 - x1);%圆心坐标
y2 = y1 + h * (y0 - y1);%圆心坐标
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk1 = y4 - (y4 - y1)/(x4 - x1) * x4;%切线与 y 轴交点
%%
dist = sqrt((x1-x01)^2+(y1-y01)^2);%距离
t1 = dist/9;
r = 20 * t1;%圆半径
h = 25/9;
x2 = x1 + h * (x01 - x1);%圆心坐标
y2 = y1 + h * (y01 - y1);%圆心坐标
```

```
x2p = (x1 + x2) / 2;%圆心坐标 1
y2p = (y1 + y2) / 2;%圆心坐标 1
R = 12.5 * t1;
a = x2 - x2p;
b = y2 - y2p;
c = (x2^2 - x2p^2 + y2^2 - y2p^2 + R^2 - r^2) / 2;
a1 = b^2 / a^2 + 1;
b1 = -2*b*c/a^2 + 2*b*x2/a - 2*y2;
c1 = c^2/a^2 + x2^2 - 2*c*x2/a + y2^2 - r^2;
y3 = (-b1+sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
y4 = (-b1-sqrt(b1^2-4*a1*c1))/(2*a1);%切点坐标
x3 = (c-b*y3)/a;
x4 = (c-b*y4)/a;
yk2 = y3 - (y3 - y1)/(x3 - x1) * x3;%切线与 y 轴交点
if yk1 > yk2
    y = [0,0];%逃离
else
    y = [(x0 + x01)/2 - x1,(y0 + y01)/2 - y1];%沿中点运动
end
end
```