

烟幕干扰弹的无人机投放策略研究

队伍编号：_____

2025 年 12 月 17 日

摘要

定点精确投放的烟幕干扰弹，可以在目标前方特定空域形成遮蔽，有效地干扰敌方导弹。本文针对无人机投放烟幕干扰弹以掩护地面目标的问题，建立了基于空气动力学和空间几何的数学模型。通过分析导弹与烟幕云团的相对运动关系，获得了不同情境下的最优投放策略，以最大化有效遮蔽时间。

针对**问题一**，我们建立了烟幕弹自由落体与扩散模型。已知无人机 *FY1* 的飞行参数，通过计算烟幕弹起爆时的空间位置与扩散半径，结合导弹 *M1* 的运动轨迹，求解出两者在时空上的交集，从而得出有效遮蔽时长。

针对**问题二**，这是一个单机单弹对单导弹的优化问题。以无人机飞行方向、速度、投放点和起爆点为决策变量，以遮蔽时间为目标函数，建立了非线性规划模型。利用遗传算法（Genetic Algorithm）进行求解，确定了使遮蔽时间最长的飞行参数。

针对**问题三**，在问题二的基础上引入多枚烟幕弹。考虑到烟幕生成的不连续性，建立了多波次干扰模型。通过规划三枚烟幕弹的时间间隔与空间分布，确立了“接力式”遮蔽策略，并将结果存入 `result1.xlsx`。

针对**问题四**和**问题五**，模型扩展为多机多弹协同对抗多枚导弹（*M1*, *M2*, *M3*）。我们将问题转化为多智能体协同任务分配（Task Assignment）与轨迹规划问题。构建了以总遮蔽效益最大化为目标的整数规划模型，给出了各架无人机的具体任务分配与投放指令。

关键词：烟幕干扰；动态几何遮蔽；多目标优化；投放策略；多机协同

目录

1	问题重述	1
1.1	问题背景	1
1.2	问题提出	1
2	问题分析	1
2.1	问题一的分析	1
2.2	问题二的分析	1
2.3	问题三的分析	2
2.4	问题四的分析	2
2.5	问题五的分析	2
3	模型假设	2
4	符号说明	2
5	模型建立	3
5.1	导弹运动模型构建	4
5.2	无人机运动模型构建	4
5.3	烟幕干扰弹的运动模型	5
5.4	烟幕弹的下沉模型	5
5.5	有效遮蔽判定模型	5
5.5.1	几何遮挡条件	5
5.5.2	点到线段距离算法	5
5.6	遮蔽时间定义	6
6	模型求解	6
6.1	问题一求解：固定参数下的遮蔽计算	6
6.1.1	参数代入与场景设定	6
6.1.2	运动轨迹计算	7
6.1.3	有效遮蔽时长计算	7
6.1.4	求解结果：	8
6.2	问题二求解：单机单弹优化	8
6.2.1	决策变量	8
6.2.2	运动状态方程	9
6.2.3	目标函数	9
6.2.4	约束条件	9
6.2.5	求解算法	9

6.3	问题三求解：多弹协同策略	10
6.3.1	决策变量	11
6.3.2	状态方程更新	11
6.3.3	目标函数	12
6.3.4	约束条件	12
6.3.5	求解算法	12
6.3.6	求解结果	13
6.4	问题四求解：多机协同优化	14
6.4.1	决策变量	14
6.4.2	状态方程	14
6.4.3	目标函数	15
6.4.4	约束条件	15
6.4.5	求解算法	15
6.4.6	求解结果	16
6.5	问题五求解：多机多目标协同拦截	17
6.5.1	决策变量体系	17
6.5.2	状态空间	17
6.5.3	目标函数	18
6.5.4	约束条件	18
6.5.5	求解算法	18
6.5.6	求解结果	19
7	模型评价	20
7.1	模型的优点	20
7.2	模型的改进方向	20

1 问题重述

1.1 问题背景

现代战争中，利用无人机投放烟幕干扰弹是保护地面重要目标的有效手段，具有成本低、效费比高等优点。烟幕弹在空中起爆后形成云团，形成遮蔽，干扰敌方导弹。本题要求在给定的战场环境下（包含来袭导弹位置、无人机位置、目标位置及物理参数），综合考虑多个物体的运动轨迹，设计无人机的飞行与投放策略，以实现最大化的遮蔽效果。

1.2 问题提出

- **问题 1：**对于给定单无人机 $FY1$ 的具体飞行参数，根据运动学建立数学模型计算出导弹 $M1$ 的有效遮蔽时长。
- **问题 2：**从问题一的特殊情况出发拓展到了一般情况，优化单无人机 $FY1$ 投放 1 枚烟幕弹的策略（方向、速度、投放点、起爆点），使遮蔽时间最大。
- **问题 3：**面对单无人机 $FY1$ 投放多枚烟幕弹的情况，规划烟雾弹的投递时序，实现总遮蔽时间的最大化。
- **问题 4：**不同于单机多枚烟雾弹的情形，情况变成了多机协同（ $FY1 - FY3$ ）各投 1 枚弹对抗 $M1$ ，需要找到其投放的最优策略。
- **问题 5：**全要素协同场景，需要找到（5 架无人机，每架最多 3 枚弹）对抗 3 枚导弹（ $M1 - M3$ ）的最优策略。

2 问题分析

2.1 问题一的分析

问题一是最简单的情景，要求计算得到给定参数下的有效遮蔽时间。首先以烟幕弹爆炸瞬间的空间位置与导弹的空间位置作为初始值，后续考虑扩散半径，结合导弹运动轨迹，求解两者的交点，从而得到有效遮蔽时长。

2.2 问题二的分析

问题二是问题一的推广，需要将无人机飞行方向、速度、投放点、起爆点作为决策变量进行优化，以最大化遮蔽时间作为目标函数，获取在满足一定的约束条件下的最佳解。实现烟幕弹的最佳投放策略。

2.3 问题三的分析

问题三是问题二的优化，需要控制好烟幕弹投放的时序，完成烟幕弹的协同投放，以实现多弹接力式的遮蔽效果，从而实现最大化有效遮蔽时间。

2.4 问题四的分析

问题四是多机协同的情况，需要考虑多架无人机在时空上的协同，实现在空间与时间上的遮蔽效果的最大化。

2.5 问题五的分析

问题五是最复杂的情景，涉及多机多弹对抗多枚导弹的情况。需要综合考虑无人机的任务分配、轨迹规划以及烟幕弹的投放时序等多个因素，建立一个综合性的优化模型，以实现整体遮蔽效益的最大化。但是直接考虑所有的决策变量，时间复杂度与空间复杂度较高，无法直接求解。因而需要对问题进行简化，将问题转化为多智能体协同任务分配与投放策略问题。

3 模型假设

为简化问题，主要假设如下：

1. 假设烟幕弹在投放后忽略空气阻力，仅受重力作用做平抛运动。
2. 假设烟幕云团形成球状且半径在有效时间内保持稳定（或按题目给定速度扩散/下沉）。
3. 假设来袭导弹做匀速直线运动，不进行机动变轨。
4. 假设雷达发现目标时刻为 $t = 0$ 。
5. 忽略无人机调整飞行方向时的转弯半径，视为瞬时变向。
6. 假设无人机、导弹、烟幕弹、烟幕云团均为质点。

4 符号说明

本文主要使用的符号定义如下表所示：

表 1: 符号说明

符号	含义	单位
M_j	第 j 枚来袭导弹 ($j = 1, 2, 3$)	-
FY_i	第 i 架无人机 ($i = 1, 2, 3, 4, 5$)	-
V_m	导弹飞行速度 (固定 300)	m/s
V_{uav}	无人机飞行速度	m/s
(x, y, z)	空间坐标位置	m
T_{mask}	有效遮蔽时间	s
t_{drop}	烟幕弹投放时刻	s
t_{boom}	烟幕弹起爆时刻	s
R_{cloud}	烟幕云团有效遮蔽半径	m

5 模型建立

根据题目描述首先建立统一的三维直角坐标系 $Oxyz$ 。以假目标中心为原点 x 轴与 y 轴位于水平面内, z 轴竖直向上, 真目标位于点 $P_T = (0, 200, h_T)$ 其中 h_T 为真目标中心高度。

第 j 枚导弹的初始位置位于点

$$P_{M_j}(0)$$

第 i 架无人机的初始位置位于点

$$P_{FY_i}(0)$$

我们可以利用 matlab 绘制初始时刻的宏观战场态势图, 如图1所示。

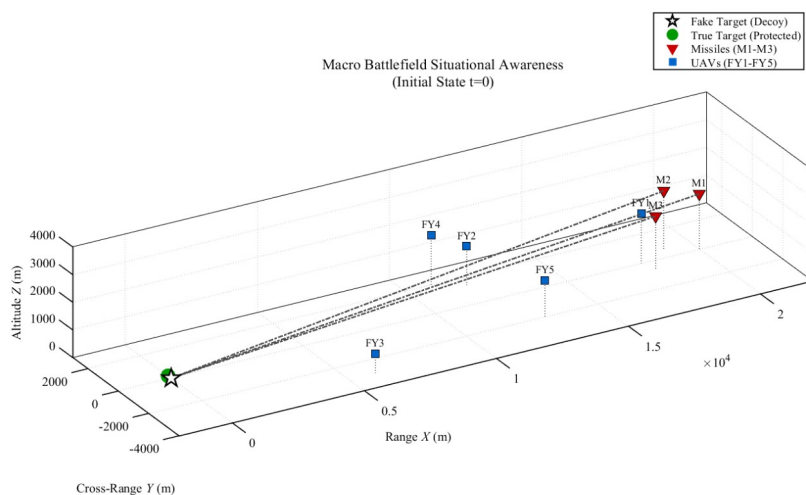


图 1: 宏观战场态势图

通过观察此图，我们可以发现，无人机群在导弹群前方按照扇形分布，形成了一个闪现拦截带，无人机 FY1 距离导弹群最近，这说明 FY1 后续可能应该优先对导弹进行拦截。同时在高度上，无人机群成梯度配置，可以形成低位前伸烟幕拦截网。

5.1 导弹运动模型构建

导弹 M_j 以速度 $V_m = 300m/s$ 直线飞向假目标 P_O ，其单位方向向量为：

$$\mathbf{n}_{M_j} = \frac{\mathbf{P}_O - \mathbf{P}_{M_j}(0)}{\|\mathbf{P}_O - \mathbf{P}_{M_j}(0)\|}$$

第 j 枚导弹初始位置为

$$\mathbf{P}_{M_j}(0)$$

导弹在任意时刻 t 的位置为：

$$\mathbf{P}_{M_j}(t) = \mathbf{P}_{M_j}(0) + V_m \mathbf{n}_{M_j}$$

5.2 无人机运动模型构建

第 i 枚无人机的初始位置为

$$\mathbf{P}_{FY_i}(0)$$

第 i 架无人机飞行速度为 v_i ，航向角为 α_i ，其速度向量表示为：

$$\mathbf{v}_{FY_i} = v_i (\cos \alpha_i, \sin \alpha_i, 0)$$

无人机在任意时刻 t 的位置为：

$$\mathbf{P}_{FY_i}(t) = \mathbf{P}_{FY_i}(0) + \mathbf{v}_{FY_i}t$$

5.3 烟幕干扰弹的运动模型

设第 k 枚烟幕弹在 $t_{drop,k}$ 时刻投放，起爆延时为 $t_{delay,k}$ ，重力加速度为 g 起爆时刻为：

$$t_{burst,k} = t_{drop,k} + t_{delay,k}$$

起爆位置为：

$$\mathbf{P}_{B,k} = \mathbf{P}_{FY_i}(t_{drop,k}) + \mathbf{v}_{FY_i}t_{delay,k} - \left(0, 0, \frac{1}{2}gt_{delay,k}^2\right)$$

5.4 烟幕弹的下沉模型

烟幕弹起爆后形成半径为 R_{cloud} 的球形云团，其中心以速度 v_{cloud} 匀速下沉：

$$\mathbf{P}_{S,k}(t) = \mathbf{P}_{B,k} - (0, 0, v_{cloud}(t - t_{p,k})), \quad t \geq t_{p,k}$$

5.5 有效遮蔽判定模型

5.5.1 几何遮挡条件

根据题目，可知无人机及导弹的初始位置离真目标（原点为 $(0, 200, 0)$ ，半径为 7，高度为 10 的圆柱体）所在位置十分遥远，而题中未提到无人机和导弹的视野，那说明其不在题目考虑范围内。我们可以合理假设导弹视野为全范围，即只要在导弹正前方的事物，都处于导弹视野范围内，那对于一个距离十分远，且相对于距离量级来说大小很小的一个真目标，便于进行计算和后续的优化问题，我们可以将其简化为一个质点，进而将遮蔽条件转化为：当烟幕云团中心到“导弹与真目标质心连线”的垂直距离小于等于烟幕半径时，视为有效遮蔽。

5.5.2 点到线段距离算法

设导弹位置为点 A ，真目标位置为点 B ，烟幕弹中心为点 P 。向量 $\overrightarrow{AB} = B - A$ ，向量 $\overrightarrow{AP} = P - A$ 。投影系数 r 计算公式为：

$$r = \frac{\overrightarrow{AP} \cdot \overrightarrow{AB}}{\|\overrightarrow{AB}\|^2}$$

距离分三种情况讨论：

$$D(t) = \begin{cases} \|\vec{AP}\| & \text{if } r \leq 0 \text{ (导弹后方)} \\ \|P - B\| & \text{if } r \geq 1 \text{ (目标后方)} \\ \frac{\|\vec{AP} \times \vec{AB}\|}{\|\vec{AB}\|} & \text{if } 0 < r < 1 \text{ (线段投影内)} \end{cases}$$

5.6 遮蔽时间定义

遮蔽时间定义为：

$$I_j(t) = \begin{cases} 1, & \text{if } D_{k,i}(t) \leq R_{cloud} \text{ and } t \geq t_{p,k} \\ 0, & \text{otherwise} \end{cases}$$

其中 $D_{k,i}(t)$ 为第 k 枚烟幕弹对第 j 枚导弹在时刻 t 的点到线段距离。

导弹 j 的总有效遮蔽时长为：

$$T_j = \int_0^{T_{end}} I_j(t) dt$$

在不同问题中，优化目标统一表示为：

$$\max T_{cover} = \sum_j T_j$$

6 模型求解

6.1 问题一求解：固定参数

问题一针对的是具体场景在给定所有的参数下 (烟幕弹的投放时间，起爆时间)，求解得出单枚烟幕弹对单枚导弹的有效遮蔽时间，需要我们利用各个物体的位置函数，结合有效遮挡判据得到有效遮挡时间。针对本题，飞行策略和投弹策略为题目给定，本题求解核心在于对已知任务的仿真模拟，所以直接使用时间步长离散化仿真即可解决本题。

6.1.1 参数代入与场景设定

根据题目条件，无人机 FY1 的初始位置为 $\mathbf{P}_{FY1}(0) = (17800, 0, 1800)$ ，飞行速度为 $v_{FY1} = 120 \text{ m/s}$ ，航向指向假目标 $\mathbf{P}_O = (0, 0, 0)$ 。烟幕干扰弹在受领任务后 $t_{drop} = 1.5 \text{ s}$ 时投放，起爆延迟时间为 $t_{delay} = 3.6 \text{ s}$ 。导弹 M1 的初始位置为 $\mathbf{P}_M(0) = (20000, 0, 2000)$ ，飞行速度为 $v_M = 300 \text{ m/s}$ ，并沿直线飞向假目标。烟幕云团半径取 $R_s = 10 \text{ m}$ 下沉速度取 $v_s = 3 \text{ m/s}$ 。

6.1.2 运动轨迹计算

由模型建立中的运动模型，可确定系统中各实体的空间位置。无人机在投放时刻的空间位置为

$$\mathbf{P}_{FY_1} = \mathbf{P}_{FY_1}(t_{drop}).$$

烟幕干扰弹的起爆时刻为

$$t_{burst} = t_{drop} + t_{delay},$$

其起爆位置为

$$\mathbf{P}_{burst} = \mathbf{P}_{FY_1}(t_{drop}) + \mathbf{v}_{FY_1} t_{delay} - (0, 0, \frac{1}{2}gt_{delay}^2).$$

起爆后，烟幕云团中心以恒定速度沿竖直方向下沉，其在任意时刻 t 的位置为

$$\mathbf{P}_S(t) = \mathbf{P}_{burst} - (0, 0, v_s(t - t_{burst})), \quad t \geq t_{burst}.$$

导弹 M1 在任意时刻 t 的位置为

$$\mathbf{P}_M(t) = \mathbf{P}_M(0) + v_M \mathbf{n}_M t,$$

其中 \mathbf{n}_M 为导弹指向假目标的单位方向向量。

6.1.3 有效遮蔽时长计算

为计算烟幕干扰弹对导弹 M1 的有效遮蔽时长，将导弹飞行全过程时间区间 $[0, T_{end}]$ 进行离散化处理。设时间步长为 Δt ，离散时间点为

$$t_k = k\Delta t, \quad k = 0, 1, \dots, N.$$

在每一离散时刻 t_k ，计算烟幕云团中心到导弹与真目标连线的最短距离 $d(t_k)$ 。当满足

$$d(t_k) \leq R_{cloud} \quad \text{且} \quad t_k \geq t_{burst}$$

时，认为该时刻烟幕对导弹 M1 实现有效遮蔽。定义遮蔽指示函数

$$I(t_k) = \begin{cases} 1, & d(t_k) \leq R_{cloud}, \quad t_k \geq t_{burst}, \\ 0, & \text{otherwise.} \end{cases}$$

则烟幕干扰弹对导弹 M1 的总有效遮蔽时长可近似表示为

$$T_{cover} \approx \sum_{k=0}^N I(t_k) \Delta t.$$

6.1.4 求解结果：

经计算,结果如下图2所示,烟幕弹起爆时刻: $t = 5.10s$, 烟幕起爆初始坐标:(17188.00, 0.00, 1736.50) 有效遮蔽时间段为 $8.1 - 9.5s$, 总时长为 $1.4s$ 。

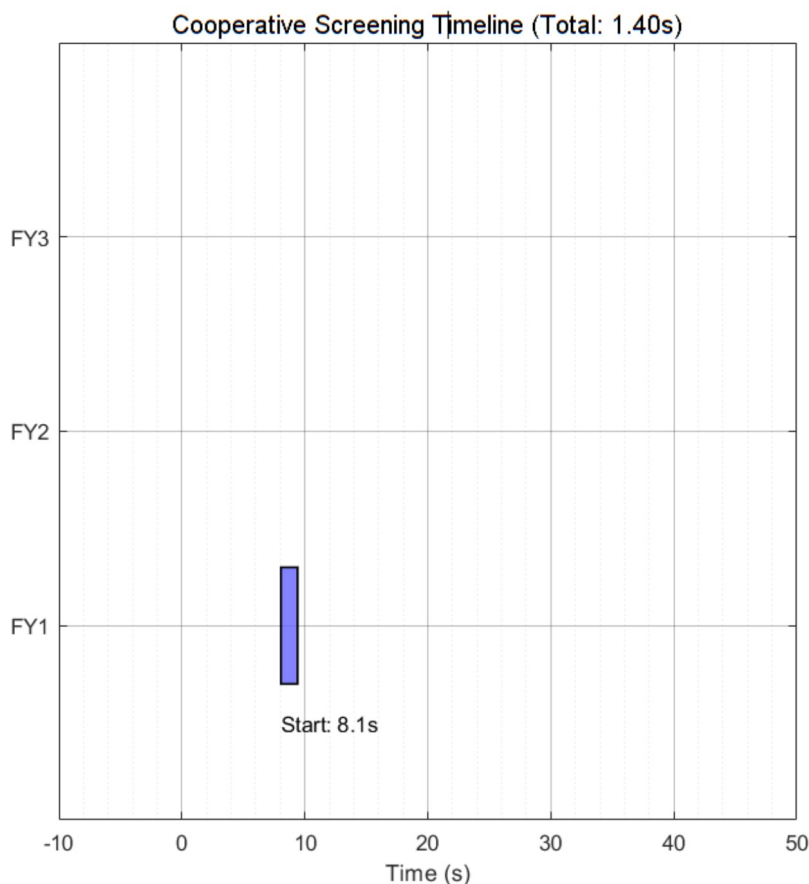


图 2: 问题一结果

6.2 问题二求解：单机单弹优化

在问题二中,无人机 FY1 的任务是通过自身的飞行参数和投弹时机,使得单枚烟雾弹对导弹 M1 的有效遮蔽时间最大化。

6.2.1 决策变量

根据题目要求,无人机一旦受领任务,其速度和航向就确定了且不再调整。因此在进行任务决策时我们需要决策以下四个变量:飞行速度 v_{FY1} , 飞行航向角 α , 投放时刻 t_{drop} , 延时起爆时间 t_{delay} , 由此定义出决策向量 X :

$$X = [v_{FY1}, \alpha, t_{drop}, t_{delay}]^T$$

6.2.2 运动状态方程

基于问题一的运动学模型，需要将无人机的速度向量参数化。无人机速度向量可表示为：

$$\mathbf{v}_{FY_1} = \begin{bmatrix} v_{FY_1} \cos \alpha \\ v_{FY_1} \sin \alpha \\ 0 \end{bmatrix}$$

投放点位置和起爆点位置更新为用 X 表示的函数：

$$\mathbf{P}_{drop}(X) = \mathbf{P}_{FY_1}(0) + \mathbf{v}_{FY_1}(v_{FY_1}, \alpha) \cdot t_{drop}$$

$$\mathbf{P}_{burst}(X) = \mathbf{P}_{drop}(X) + \mathbf{v}_{FY_1}(v_{FY_1}, \alpha) \cdot t_{delay} + \left[0, 0, -\frac{1}{2}gt_{delay}^2 \right]^T$$

6.2.3 目标函数

目标是最大化有效遮蔽时长，利用问题一中定义的遮蔽判定指示函数得出目标函数：

$$\max J(X) = T_{cover} = \int_0^{T_{end}} I(t, X) dt$$

6.2.4 约束条件

根据题目和对于模型速度特征的分析，约束条件如下：速度约束：

$$70 \leq v_{FY_1} \leq 140$$

时间约束：

$$0 < t_{drop} \leq 12, 1 \leq t_{delay} \leq 8$$

高度约束：

$$z_{burst}(X) > 0$$

6.2.5 求解算法

由于目标函数涉及复杂的几何运动及分段逻辑，具有高度的非线性和非凸性，且决策变量相对于一般的单变量优化问题具有高自由度。所以为了解决此类问题，本问题采用双层优化策略，外层使用网格扫描，内层使用粒子群优化算法。

算法整体架构 外层循环（针对速度，因为速度对优化结果的影响最大）：将速度在 $[70, 140]$ 区间内以 0.5m/s 为步长进行离散化扫描。内层循环（针对剩余参数）：针对每一个固定的速度，使用 PSO 算法寻找最优组合。这样分层寻优的好处是将优化问题维度从 4 维降至 3 维，降低了搜索难度，降低问题陷入局部最优的概率。

粒子群算法设计 在内层针对剩余决策变量进行寻优。设置粒子群规模为 $N=30$ ；粒子速度与位置更新公式：

$$v_i^{k+1} = wv_i^k + c_1r_1(p_{best,i} - x_i^k) + c_2r_2(g_{best} - x_i^k)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}$$

适应度函数：

$$F(x) = -T_{cover}(x)$$

求解结果： 经过计算，得到最优投放策略如下：

无人机以 196.91° 的航向角度， 72m/s 的速度， 0s 时刻投放， 2.4991s 后起爆，有效遮蔽时长为 4.738s 如下图3所示。

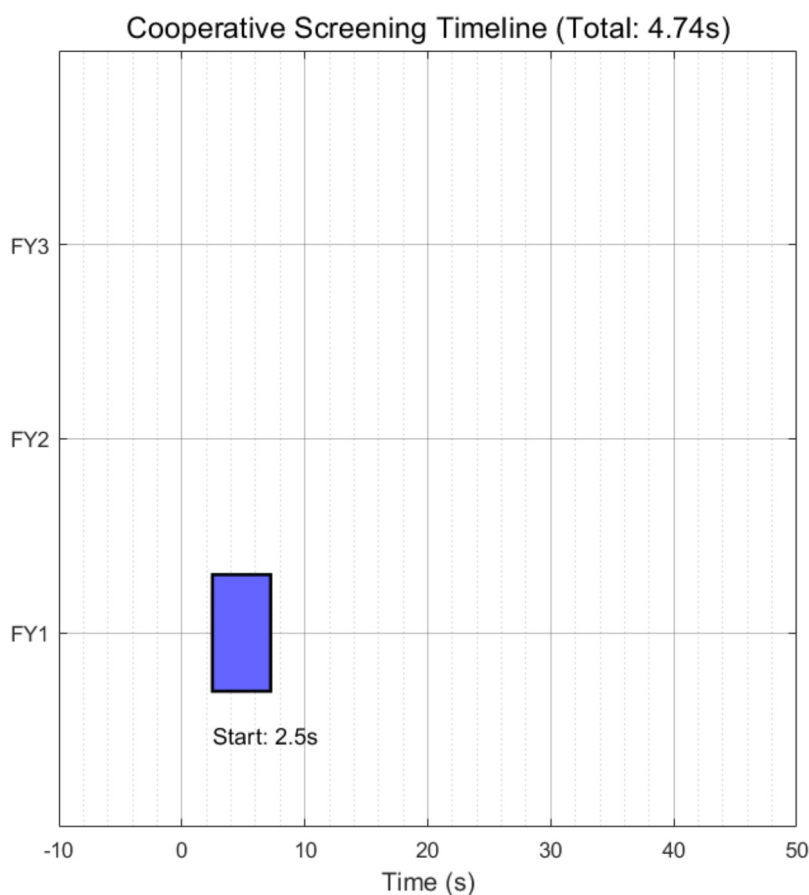


图 3: 问题二结果

6.3 问题三求解：多弹协同策略

在本题中，无人机 FY1 需要连续投放 3 枚烟幕弹以对导弹 M1 实施干扰。与投放一枚烟幕弹不同，三枚烟幕弹的投弹策略涉及时序上的配合，核心问题在对多参数的优化使得烟幕弹的遮蔽时间最大

6.3.1 决策变量

根据题目要求，无人机一旦受领任务，其速度和航向就确定了且不再调整。因此在
进行任务决策时我们需要决策以下八个变量：

飞行速度

$$v_{FY1}$$

飞行航向角

$$\alpha$$

投放时刻 1

$$t_{drop,1}$$

投放时刻 2

$$t_{drop,2}$$

投放时刻 3

$$t_{drop,3}$$

延时起爆时间 1

$$t_{delay,1}$$

延时起爆时间 2

$$t_{delay,2}$$

延时起爆时间 3

$$t_{delay,3}$$

由此定义出决策向量 \mathbf{X} :

$$\mathbf{X} = [v_{FY1}, \alpha, t_{drop,1}, t_{drop,2}, t_{drop,3}, t_{delay,1}, t_{delay,2}, t_{delay,3}]^T$$

6.3.2 状态方程更新

基于问题一的运动学模型，第 k 枚烟幕弹的运动状态由下式描述 1. 投放点坐标：

$$P_{drop,k}(\mathbf{X}) = B_0 + \mathbf{v}_{FY1}(v_{FY1}, \alpha) \cdot t_{drop,k}$$

2. 起爆点坐标

$$P_{burst,k}(\mathbf{X}) = P_{drop,k}(\mathbf{X}) + \mathbf{v}_{FY1}(v_{FY1}, \alpha) \cdot t_{delay,k} + \left[0, 0, -\frac{1}{2}gt_{delay,k}^2 \right]^T$$

3. 烟幕云团中心轨迹：起爆时刻 $t_{burst,k} = t_{drop,k} + t_{delay,k}$ 后，第 k 个云团中心位置为

$$P_{S,k}(t) = P_{burst,k} - [0, 0, v_{sink} \cdot (t - t_{burst,k})]^T, \quad t \in [t_{burst,k}, t_{burst,k} + 20]$$

6.3.3 目标函数

由于 3 个烟幕团可能同时存在并产生遮挡效果，总遮蔽时间不能简单求和，而应计算时间轴上的并集。定义第 k 枚烟幕弹在 t 时刻的遮蔽指示函数 $I_k(t; \mathbf{X})$:

$$I_k(t; \mathbf{X}) = \begin{cases} 1, & \text{if } D_k(t) \leq R_{cloud} \text{ and } t \in [t_{burst,k}, t_{burst,k} + 20] \\ 0, & \text{otherwise} \end{cases}$$

其中 $D_k(t)$ 为第 k 个烟幕中心到“导弹-目标”视线的距离（计算方法同问题一）。

系统在 t 时刻的遮蔽状态是所有单个烟幕弹的逻辑或

$$I_{total}(t; \mathbf{X}) = \max\{I_1(t), I_2(t), I_3(t)\}$$

最终优化目标为最大化总有效遮蔽时长 T_{cover} :

$$\max J(\mathbf{X}) = T_{cover} = \int_0^{T_{end}} I_{total}(t; \mathbf{X}) dt$$

6.3.4 约束条件

根据题目和对于模型速度特征的分析，约束条件如下：除了满足问题二中的飞行与高度约束外，本问必须严格满足多枚弹之间的操作间隔约束：

投弹间隔约束：题目要求每架无人机投放两枚烟幕弹至少间隔 1s。

$$t_{drop,k+1} - t_{drop,k} \geq 1, \quad k = 1, 2$$

基本边界约束：

$$\begin{cases} 70 \leq v_{FY_1} \leq 140 \\ 0 < t_{drop,1} < t_{drop,2} < t_{drop,3} \leq 12 \\ 1 \leq t_{delay,k} \leq 8, \quad k = 1, 2, 3 \end{cases}$$

高度约束：

$$z_{burst,k}(\mathbf{X}) > 0, \quad k = 1, 2, 3$$

6.3.5 求解算法

针对决策变量维数增加（由 4 维增至 8 维）且包含时序逻辑约束的问题，传统的优化算法容易陷入停滞，因此我们采用引入灾变机制的改进粒子群算法求解。

算法整体架构 不再采用网格扫描，而是直接在 8 维解空间内进行全局寻优。算法通过监测种群的进化状态，动态调整搜索策略，平衡全局探索与局部开发能力。

改进粒子群算法设计

- 粒子编码与约束处理：

虽然决策变量定义为绝对时刻，但在算法实现中，为了自动满足间隔约束，对时间参数采用增量编码 ($\Delta t_{12}, \Delta t_{23}$) 进行搜索，解码时再转换为绝对时间 t_{drop} 。

- 灾变重启机制：

针对多波次协同容易陷入局部最优的问题，引入”灾变”策略

停滞检测：设置停滞计数器 $stagnation_counter$ ，若全局最优解 G_{best} 连续一定代数未更新，则判定算法陷入停滞。

灾变操作：触发灾变，保留种群中前 10% 的精英粒子（历史最优个体），对其余 90% 的粒子进行完全重置（位置与速度重新随机化）。该机制能有效打破种群的聚集状态，跳出局部极值陷阱，寻找更优的多弹协同策略。

- 适应度函数：对时间轴进行离散化扫描，计算每一时刻系统总遮蔽状态 $I_{sys}(t)$ ，积分得到总有效时长，取负值作为适应度函数 $F(\mathbf{X}) = -T_{cover}$ 进行极小化求解。

6.3.6 求解结果

根据上述的流程，我们采用灾变机制的改进粒子群算法得到无人机 FY1 投放 3 枚烟幕弹干扰导弹 M1 的最优策略见下图4

对应的最大有效遮挡时间是 12.54241s

无人机航向方向角为:180.67°

无人机速度大小为:95m/s。

三枚烟幕弹的具体投放策略见下表2。

表 2: Problem 3 最终策略

参数	数值
最大有效遮蔽时长	12.54241 s
无人机速度	95 m/s
无人机航向	3.1533 rad (180.67°)
烟弹投放与起爆时刻	
烟弹 1 投放时刻	0.0000 s
烟弹 1 起爆时刻	0.0000 s
烟弹 2 投放时刻	2.7509 s
烟弹 2 起爆时刻	6.4326 s
烟弹 3 投放时刻	8.0057 s
烟弹 3 起爆时刻	13.3706 s

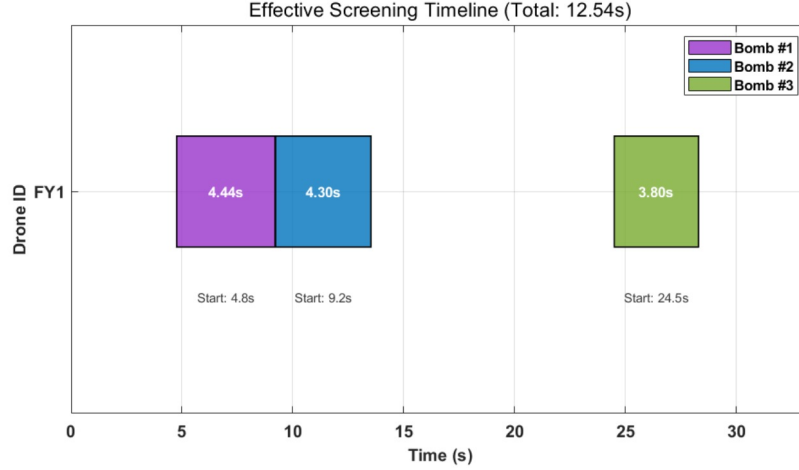


图 4: Problem 3 结果

6.4 问题四求解：多机协同优化

在问题四中，任务变更为 FY_1 FY_2 FY_3 三架无人机共同拦截导弹 $M1$ ，每架无人机投放一枚烟幕弹，核心问题在于协调三架处于不同空间位置的无人机的投弹策略，使得烟幕弹的遮蔽时间最大。

6.4.1 决策变量

根据题目要求，无人机一旦受领任务，其速度和航向就确定了且不再调整。因此在进行任务决策时我们需要决策 12 维变量：系统包含 3 个独立的运动实体（无人机）。定义 $i \in 1, 2, 3$ 分别对应 FY_1, FY_2, FY_3 。每个实体的决策向量 \mathbf{u}_i 包含 4 个参数：

$$\mathbf{u}_i = [v_i, \alpha_i, t_{drop,i}, t_{delay,i}]$$

全局决策向量 \mathbf{X} 维 12 维向量：

$$\mathbf{X} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = [v1, 1, t_{drop, 1}, t_{delay, 1}, \dots, v3, 3, t_{drop, 3}, t_{delay, 3}]^T$$

6.4.2 状态方程

各无人机的初始位置 $\mathbf{Pos}_{UAV,i}$ 不同，第 i 架无人机投放的烟幕弹在 t 刻的中心位置 $P_{S,i}(t)$ 更新公式如下投放点坐标：

$$P_{drop,i} = \mathbf{Pos}_{UAV,i} + \mathbf{v}_i(v_i, \alpha_i) \cdot t_{drop,i}$$

起爆点坐标

$$P_{burst,i} = P_{drop,i} + \mathbf{v}_i \cdot t_{delay,i} + \mathbf{G}(t_{delay,i})$$

其中 $\mathbf{G}(t) = [0, 0, -\frac{1}{2}gt^2]^T$ 为重力项。

烟幕云团中心轨迹：

$$P_{S,i}(t) = P_{burst,i} - [0, 0, v_{sink} \cdot (t - t_{burst,i})]^T$$

6.4.3 目标函数

由于 3 个烟幕团可能同时存在并产生遮挡效果，总遮蔽时间不能简单求和，而应计算时间轴上的并集。定义第 i 个烟幕云团对目标的遮蔽指示函数 $I_i(t)$ ：

$$I_i(t; \mathbf{X}) = \begin{cases} 1, & \text{if } \text{dist}(P_{S,i}(t), \text{Line}_{M1-Target}) \leq R_{cloud} \\ 0, & \text{otherwise} \end{cases}$$

系统总遮蔽状态为各子状态的逻辑或：

$$I_{sys}(t) = \max_{i=1,2,3} I_i(t)$$

最终优化目标函数定义为最小化代价 $J(\mathbf{X})$ ：

$$\min J(\mathbf{X}) = - \left(\int_0^{T_{end}} I_{sys}(t) dt \right) + \lambda \cdot \sum_{i=1}^3 \max(d_{min,i}, R_{cloud})$$

其中：第一项为总遮蔽时长（取负求最小），第二项为惩罚项， $d_{min,i}$ 为第 i 枚烟幕弹在其生命周期内距离视线的最近距离， $\lambda = 0.1$ 为权重系数。当烟幕完全脱靶时，该项通过惩罚距离迫使解向视线靠拢。

6.4.4 约束条件

速度约束：

$$70 \leq v_i \leq 140, \quad \forall i$$

时间窗约束：我们通过物理知识可知，遮蔽的理想状态是烟幕弹之间无缝接力，所以设置时间窗口约束，减小搜索空间：

$$t_{drop,1} \leq 15, \quad t_{drop,2} \leq 35, \quad t_{drop,3} \leq 55$$

起爆延时约束：

$$1 \leq t_{delay,i} \leq 8$$

高度约束：

$$z_{burst,k}(\mathbf{X}) > 0, \quad k = 1, 2, 3$$

6.4.5 求解算法

由于决策变量维数增加为 12 维，而且各维度之间差异较大，所以第四问我们选择鲁棒性强的差分进化算法（DE）求解

算法流程设计

- 种群初始化：生成 $NP=200$ 个个体。针对航向角 α_i ，基于各无人机到目标的初始视线角 $\theta_{base,i}$ 进行正态分布初始化，范围控制在 $\theta_{base,i} \pm 45^\circ$ 内，提高搜索效率
- 变异：采用 DE/rand/1 策略生成变异向量 \mathbf{V}_g ：

$$\mathbf{V}_{i,g} = \mathbf{X}_{r1,g} + F \cdot (\mathbf{X}_{r2,g} - \mathbf{X}_{r3,g})$$

其中缩放因子 F 采用自适应策略，随迭代次数从 0.5 线性衰减，平衡全局探索与局部开发。

- 交叉：对变异向量与目标向量进行二项式交叉，生成试验向量 $\mathbf{U}_{i,g}$ 。交叉概率 $CR = 0.9$ 。
- 选择：计算试验向量与目标向量的适应度，选择适应度更优的个体进入下一代。采用贪婪策略，若试验向量的适应度优于目标向量，则在下一代中保留试验向量：

$$\mathbf{X}_{i,g+1} = \begin{cases} \mathbf{U}_{i,g}, & \text{if } J(\mathbf{U}_{i,g}) < J(\mathbf{X}_{i,g}) \\ \mathbf{X}_{i,g}, & \text{otherwise} \end{cases}$$

- 终止条件：最大迭代次数 $G_{max} = 800$ 或适应度不再显著提升

6.4.6 求解结果

我们通过差分进化算法（DE）用 matlab 求解得到了最大的总遮蔽有效时间 15s，并且烟幕弹的遮蔽时间衔接良好，提供较好的遮蔽效果。对应的三机协同投放策略如下表6.4.6所示，效果如下图5所示。

表 3: 问题 4 三机协同立体封锁最终策略

机号	初始位置 (m)	速度 (m/s)	航向 ($^\circ$)	投放时刻 (s)	延时 (s)	起爆时刻 (s)
FY1	(17800, 0, 1800)	78.65	178.42	0.67	2.87	3.55
FY2	(12000, 1400, 1400)	119.78	-129.29	6.69	7.84	14.53
FY3	(6000, -3000, 700)	135.11	106.94	18.00	6.09	24.09

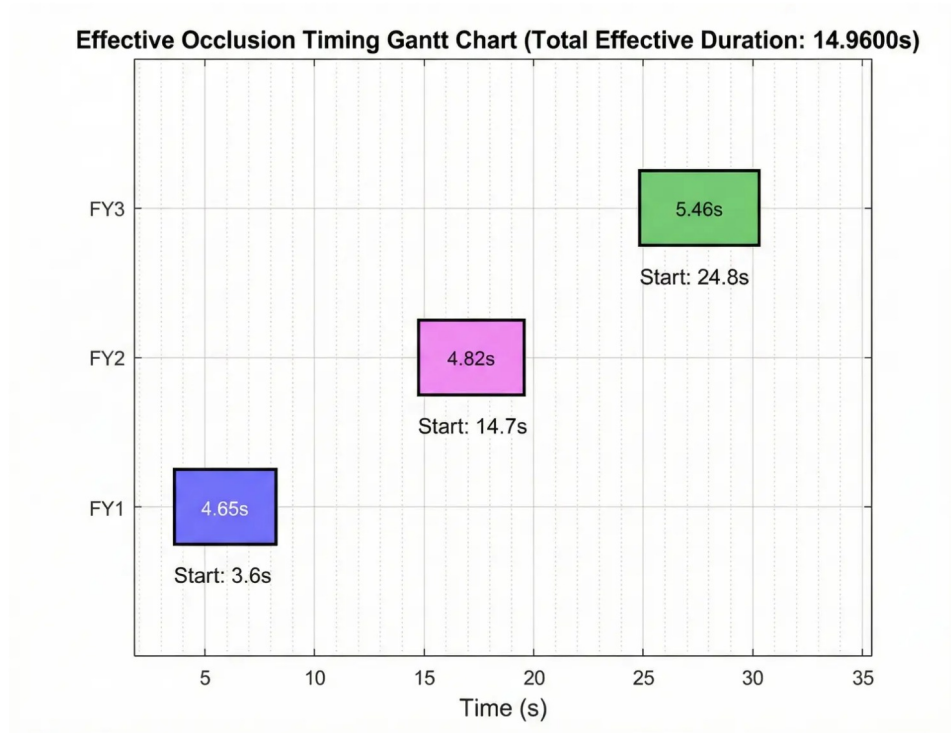


图 5: 问题 4 三机协同立体封锁结果

6.5 问题五求解：多机多目标协同拦截

在问题五中，作战场景扩展为 5 架无人机 $FY_1 - FY_5$ 协同拦截 3 枚来袭导弹 $M_1 - M_3$ 。每架无人机最多可携带 3 枚烟幕弹。问题的核心在于如何在复杂的空间分布下，合理分配无人机的飞行策略和弹药资源，使得对所有导弹的总有效遮蔽时间最大化。

6.5.1 决策变量体系

考虑到问题规模显著扩大，且受到前几问优化算法的启发，我们将决策变量分为战略层（飞行参数）和战术层（投弹参数）。战略决策变量：定义全局飞行决策向量，包含 5 架无人机的速度和航向：

$$X_{fly} = [v_1, \alpha_1, v_2, \alpha_2, \dots, v_5, \alpha_5]^T$$

战术决策变量：对于第 i 架无人机，其携带的第 k 枚烟幕弹 $k = 1, 2, 3$ 的投放参数由算法在内层生成，不直接作为外层优化变量，但需满足物理约束：

$$u_{i,k} = \begin{bmatrix} t_{drop}^{(i,k)} \\ t_{delay}^{(i,k)} \end{bmatrix}$$

6.5.2 状态空间

多导弹运动模型的建立参考第一问，因为每个导弹的飞行特性为一致，只有初始位置不同。多无人机运动模型亦可由之前类比得到。

6.5.3 目标函数

由于存在 3 个独立的来袭目标，系统总效益定义为所有导弹被有效遮蔽时长的总和：

$$S_j(t) = \bigvee_{i=1}^5 \bigvee_{k=1}^3 I_{i,k,j}(t)$$

最终优化目标为最大化总效益 J：

$$\max J(X_{fly}) = \sum_{j=1}^3 \left(\int_0^{T_{end}} S_j(t) dt \right)$$

6.5.4 约束条件

最大载弹量约束：每架无人机投放数量 $N \leq 3$ ；

投弹间隔约束：

$$|t_{drop}^{(i,k)} - t_{drop}^{(i,k-1)}| \geq 1, \forall k > 1$$

飞行边界约束：

$$70 \leq v_i \leq 140, 0 \leq \alpha_i \leq 2\pi$$

6.5.5 求解算法

针对 5 机 3 弹的高维组合优化问题，依旧采用双层优化结构。外层使用遗传算法搜索无人机的最优飞行参数，内层使用贪心启发式算法快速计算给定航迹下的最优投弹方案。

- 外层：遗传算法全局寻优将 5 架无人机的速度和航向编码为一条染色体。采用实数编码，操作算子采用迷你二进制交叉和多项式变异。实现编码 X_{fly} ，则适应度函数为：

$$Fitness(X_{fly}) = -J(X_{fly})$$

- 内层：贪心策略解算投弹方案对于种群中每一个体确定的飞行轨迹，内层算法通过以下步骤计算适应度：

Step 1: 生成候选打击集遍历第 i 架无人机飞行路径上的离散时间点 t ，对于每一个 t ，遍历可能的延时起爆时间，检测生成的烟幕云团能否拦截 M1, M2, M3 中的任意一枚。若能拦截，记录该候选方案。

Step 2: 贪心选择对候选方案集按遮蔽时长降序排列。依次选择遮蔽贡献最大的方案加入最终策略，需满足：该无人机已选弹药数 < 3 ；与该无人机已选方案的时间间隔 $\geq 1s$ 。

Step 3: 计算并集效益汇总所有无人机的投弹方案，分别计算对 M1, M2, M3 的遮蔽时间区间并集，求和得到总分。此方法的优势在于将复杂的时序配合问题解耦：外层负责“站位”（规划航线），内层负责“输出”（寻找最佳开火时机），在保证解的质量的同时大幅降低了计算复杂度。

6.5.6 求解结果

通过该算法求解，得到 5 机 3 弹协同拦截 3 枚导弹的最优策略如下表6.5.6所示，总有效遮蔽时间为 28.763s，效果如图6所示。

表 4: 五机协同的详细方案

Drone	BombID	DropTime (s)	ExpTime (s)	Target	CoverDuration (s)	Interval
FY1: 88.19 m/s, 178.35°(3.11 rad)						
FY1	#1	0.00	3.00	M1	3.80	[3.0–6.8]
FY1	#2	1.00	4.00	M	3.80	[4.6–8.4]
FY2: 110.93 m/s, 283.65°(4.95 rad)						
FY2	#1	6.00	9.00	M2	4.00	[9.0–13.0]
FY2	#2	8.00	13.00	M1	0.80	[24.8–25.6]
FY3: 108.05 m/s, 121.72°(2.12 rad)						
FY3	#1	28.00	35.00	M2	3.40	[37.8–41.2]
FY3	#2	25.00	32.00	M3	3.00	[35.6–38.6]
FY3	#3	26.00	33.00	M	1.60	[49.4–51.0]
FY4: 135.80 m/s, 262.62°(4.58 rad)						
FY4	#1	1.00	12.00	M2	4.40	[15.0–19.4]
FY5: 134.90 m/s, 123.38°(2.15 rad)						
FY5	#1	13.00	18.00	M1	4.00	[19.6–23.6]

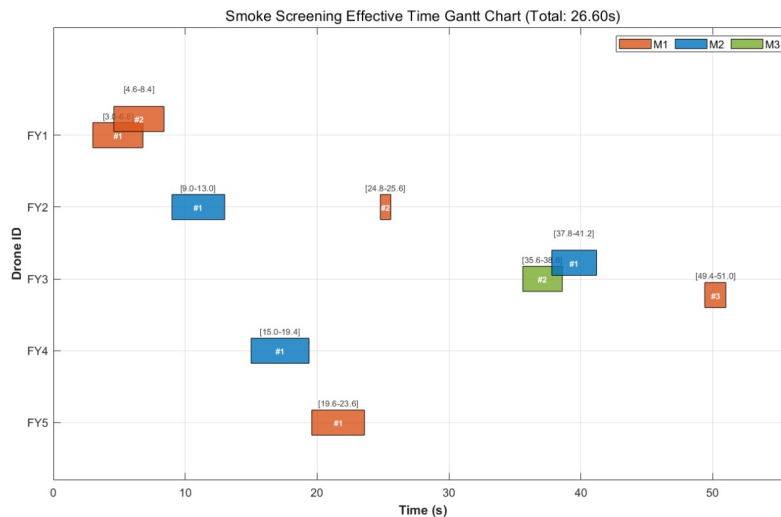


图 6: 五机协同的最优方案

7 模型评价

7.1 模型的优点

- 考虑了烟幕弹的物理沉降特性，符合实际战场环境。
- 算法具有较强的通用性，可扩展至更多数量的无人机集群。

7.2 模型的改进方向

- 当前模型假设导弹不机动，未来可增加导弹末端机动的对抗策略。

参考文献

- [1] 作者. 题目. 期刊名, 年份, 卷 (期): 页码.
- [2] 张三. 无人机协同作战理论. 北京: 科学出版社, 2020.
- [3] CUMCM 组委会. 202x 年全国大学生数学建模竞赛赛题.

附录：主要代码

MATLAB 轨迹仿真代码

Listing 1: Problem 1

```

1  clc; clear; close all;
2  %% 1. 参数设置
3  % 烟幕弹参数
4  R_smoke = 10;           % 有效遮蔽半径（阈值）
5  Time_smoke_last = 25;   % 延长一点仿真时间以便看全曲线
6  V_smoke_sink = 3;       % 烟雾团下沉速度
7  % 物理常数
8  g = 9.8;
9  % 目标位置
10 Pos_FakeTarget = [0, 0, 0];           % 假目标（导弹飞行的目的地）
11 Pos_TrueTarget_Center = [0, 200, 5]; % 真目标（我们要保护的對象）
12 %% 2. 计算烟幕弹起爆初始位置
13 % FY1 初始状态
14 Pos_FY1_0 = [17800, 0, 1800];
15 V_FY1 = 120;
16 % 时间节点
17 t_drop = 1.5;           % 投放时刻
18 t_delay = 3.6;          % 延时时长
19 t_pop = t_drop + t_delay; % 起爆时刻（5.1s）
20 % 1. 投放点位置（FY1 飞行 1.5s）
21 Pos_Drop = Pos_FY1_0 + [-1, 0, 0] * V_FY1 * t_drop;
22 % 2. 起爆点初始位置（平抛运动 3.6s）
23 Delta_X = -1 * V_FY1 * t_delay;
24 Delta_Z = -0.5 * g * t_delay^2;
25 Pos_Smoke_Init = Pos_Drop + [Delta_X, 0, Delta_Z];
26 fprintf('烟幕弹起爆时刻: t = %.2f s\n', t_pop);
27 fprintf('烟幕起爆初始坐标: (%.2f, %.2f, %.2f)\n', Pos_Smoke_Init);
28 %% 3. 计算有效遮蔽时长（并记录数据用于画图）
29 % M1 初始状态
30 Pos_M1_0 = [20000, 0, 2000];
31 V_M1 = 300;
32 Vec_M1 = Pos_FakeTarget - Pos_M1_0; % 方向指向假目标
33 Dist_M1_Total = norm(Vec_M1);
34 Dir_M1 = Vec_M1 / Dist_M1_Total;
35 % 时间积分设置

```



```

36 dt = 0.05; % 稍微调大一点点dt, 画图不需要太密集
37 valid_time = 0;
38 % --- 初始化数据记录数组 ---
39 t_log = []; % 记录时间
40 dist_log = []; % 记录距离
41 fprintf('正在模拟动态遮挡过程...\n');
42 % 从 t=0 开始记录导弹位置, 但 t<t_pop 时距离设为 NaN 或 到初始点的距
    离
43 Total_Sim_Time = 30; % 总仿真时长 (对应画图横坐标 0-30s)
44 for t_current = 0 : dt : Total_Sim_Time
45
46     % A. 更新导弹位置 (一直飞)
47     dist_flown = V_M1 * t_current;
48     if dist_flown >= Dist_M1_Total
49         Current_M1_Pos = Pos_FakeTarget; % 已经击中假目标
50     else
51         Current_M1_Pos = Pos_M1_0 + Dir_M1 * dist_flown;
52     end
53
54     % B. 更新烟雾位置 & 计算距离
55     % 只有在时间超过起爆时间后, 烟雾才存在
56     if t_current >= t_pop
57         t_relative = t_current - t_pop;
58
59         % 烟雾下沉
60         if t_relative <= Time_smoke_last
61             Sink_Dist = V_smoke_sink * t_relative;
62             Current_Smoke_Pos = Pos_Smoke_Init - [0, 0, Sink_Dist];
63
64             % --- 计算遮挡距离 (核心算法) ---
65             P1 = Current_M1_Pos;
66             P2 = Pos_TrueTarget_Center;
67             Q = Current_Smoke_Pos;
68
69             v = P2 - P1;
70             w = Q - P1;
71             c1 = dot(w, v);
72             c2 = dot(v, v);
73
74             if c1 <= 0

```

```

75         dist = norm(Q - P1);
76     elseif c2 <= c1
77         dist = norm(Q - P2);
78     else
79         b = c1 / c2;
80         Pb = P1 + b * v;
81         dist = norm(Q - Pb);
82     end
83
84     % 统计有效时间
85     if dist <= R_smoke
86         valid_time = valid_time + dt;
87     end
88
89     else
90         % 烟雾消失了
91         dist = NaN;
92     end
93 else
94     % 还没起爆，计算导弹到"未来起爆点"的距离
95     dist = norm(Pos_Smoke_Init - Current_M1_Pos);
96 end
97
98 % C. 记录数据
99 t_log = [t_log, t_current];
100 dist_log = [dist_log, dist];
101 end
102 fprintf('-----\n');
103 fprintf('>>>最终结果：有效遮蔽时长为：%.4f秒<<<\n', valid_time);
104 %% 4. 绘图（完全复刻之前的样式）
105 figure('Color', 'w', 'Name', '烟幕干扰弹有效遮蔽时间分析');
106 hold on; grid on; box on;
107 % --- A. 准备填充区域数据 ---
108 % 找到距离小于阈值且时间大于起爆时间的索引
109 mask = (dist_log <= R_smoke) & (t_log >= t_pop);
110 if any(mask)
111     % 提取需要填充的时间段和距离
112     t_fill_data = t_log(mask);
113     dist_fill_data = dist_log(mask);
114

```

```

115 % 构造多边形：上边缘是阈值线，下边缘是实际曲线
116 % 顶点顺序：[左上 -> 曲线点集 -> 右上]
117 X_poly = [t_fill_data, fliplr(t_fill_data)];
118 Y_poly = [dist_fill_data, ones(size(dist_fill_data)) * R_smoke];
119
120 % 绘制淡黄色填充
121 h_area = fill(X_poly, Y_poly, [1 1 0.8], 'EdgeColor', 'none', '
    FaceAlpha', 0.5);
122
123 % 找到有效结束时间（遮蔽区间的最后一个时刻）
124 t_end_effective = t_fill_data(end);
125 else
126     t_end_effective = t_pop; % 如果没遮住，结束时间就标在起爆点
127     h_area = plot(NaN, NaN, 'y'); % 占位符防止legend报错
128 end
129 % --- B. 绘制主要曲线 ---
130 h_dist = plot(t_log, dist_log, 'b-', 'LineWidth', 2);
131 % --- C. 绘制辅助线（阈值、起爆、结束） ---
132 % 1. 阈值红虚线
133 h_thresh = plot([0, 30], [R_smoke, R_smoke], 'r--', 'LineWidth', 1);
134 % 2. 起爆时间绿虚线
135 h_start = xline(t_pop, 'g--', 'LineWidth', 1.5);
136 % 3. 有效结束绿虚线（如果有遮蔽的话）
137 if any(mask)
138     h_end = xline(t_end_effective, 'g--', 'LineWidth', 1.5);
139     % 标注文字：有效结束
140     text(t_end_effective + 0.5, 60, '有效结束', 'Color', 'g', '
        Rotation', 90, 'FontSize', 10);
141 else
142     h_end = plot(NaN, NaN, 'g--'); % 占位
143 end
144 % --- D. 添加文字标注 ---
145 % 起爆时间
146 text(t_pop - 0.8, 60, '起爆时间', 'Color', 'g', 'Rotation', 90, '
    FontSize', 10);
147 text(25, R_smoke + 2, ['阈值' num2str(R_smoke) 'm'], 'Color', 'r', '
    FontSize', 10);
148 % --- E. 图例与坐标轴设置 ---
149 title('烟幕干扰弹有效遮蔽时间分析', 'FontSize', 12);
150 xlabel('时间 (s)', 'FontSize', 11);

```

```

151 ylabel('到导弹路径的垂直距离 $\perp$ (m)', 'FontSize', 11);
152 xlim([0, 30]);
153 ylim([0, 70]); % 根据你的图片要求定在70, 如果数据很大可以改大
154 legend([h_dist, h_thresh, h_start, h_end, h_area], ...
155         {'距离曲线', '阈值', '起爆时间', '有效结束', '有效遮蔽区间'}, ...
156         'Location', 'north', 'Orientation', 'vertical');
157 hold off;

```

Listing 2: Problem 2

```

1  function Full_Range_PSO_Sweep()
2  clc; close all;
3
4  %% 1. 场景基础参数
5  Env.g = 9.8;
6  Env.Pos_FakeTarget = [0, 0, 0];
7  Env.Pos_TrueTarget = [0, 200, 5];
8  Env.Pos_M1_Init = [20000, 0, 2000];
9  Env.V_M1 = 300;
10 Env.Vec_M = Env.Pos_FakeTarget - Env.Pos_M1_Init;
11 Env.Dir_M1 = Env.Vec_M / norm(Env.Vec_M);
12 Env.Dist_Total_M1 = norm(Env.Vec_M);
13 Env.Pos_FY1_Init = [17800, 0, 1800];
14 Env.V_smoke_sink = 3;
15 Env.R_smoke = 10;
16 Env.Time_smoke_last = 25;
17
18 % === 仿真精度设置 ===
19 % 在PSO搜索阶段使用 1ms 精度, 足够区分优劣且速度可控
20 Env.dt = 0.001;
21
22 %% 2. 扫描设置
23 v_scan_list = 70 : 0.5 : 140; % 您的要求: 70-140, 步长0.5
24 num_scan = length(v_scan_list);
25
26 % 结果存储
27 results_score = zeros(num_scan, 1);
28 results_params = zeros(num_scan, 3); % [航向, 投放, 延时]
29
30 fprintf('=====\n
    ');

```

```

31     fprintf('        FY1_全速域_PSO_暴力扫描_(Grid-PSO)\n');
32     fprintf('        速度: 70-140m/s | 步长: 0.5m/s | 核心: PSO\n');
33     fprintf('=====\\n
        ');
34
35     try
36         if isempty(gcp('nocreate')), parpool; end
37         fprintf('>>>_并行计算池已就绪, 全速运算中...\n');
38     catch
39         fprintf('>>>_并行启动失败, 切换为单核运算...\n');
40     end
41
42     timer_start = tic;
43
44     %% 3. 并行扫描循环
45     parfor i = 1 : num_scan
46         v_curr = v_scan_list(i);
47
48         % --- PSO 配置 ---
49         % 在固定的速度下, 寻找最佳的 [航向, 投放, 延时]
50         % 变量维度: 3
51         % 搜索范围:
52         %   航向: 基准 +/- 30度
53         %   投放: 0 - 12秒
54         %   延时: 1 - 8秒
55
56         % 计算基准航向
57         Vec_Base = Env.Pos_TrueTarget - Env.Pos_FY1_Init;
58         Base_Angle = rad2deg(atan2(Vec_Base(2), Vec_Base(1)));
59
60         LB = [Base_Angle-30, 0, 1.0];
61         UB = [Base_Angle+30, 12, 8.0];
62
63         % 运行内置的微型 PSO
64         % 粒子数: 30 (对3维问题足够)
65         % 迭代: 50
66         [best_x, best_val] = Run_Micro_PSO(v_curr, LB, UB, Env);
67
68         % 记录
69         results_score(i) = best_val;

```

```

70         results_params(i, :) = best_x;
71
72         if mod(i, 10) == 0
73             fprintf('#####...完成速度%.1f_m/s(最佳遮蔽:%.4fs)\n',
74                     v_curr, best_val);
75         end
76     end
77
78     total_time = toc(timer_start);
79     fprintf('\n>>>扫描完成!总耗时:%.2f秒\n', total_time);
80
81     %% 4. 结果分析与可视化
82     [max_score, idx_best] = max(results_score);
83     best_v = v_scan_list(idx_best);
84     best_p = results_params(idx_best, :);
85
86     fprintf('\n
87     =====\n');
88     fprintf('#####全局最优解(Global Optimum)\n');
89     fprintf('=====
90     ');
91
92     fprintf('>>>冠军速度:%.1f_m/s\n', best_v);
93     fprintf('>>>极限遮蔽:%.5f秒\n', max_score);
94     fprintf('>>>战术参数:航向%.2f度|投放%.4fs|延时%.4fs\n',
95             best_p(1), best_p(2), best_p(3));
96
97     % --- 绘制“速度-性能”全景图 ---
98     figure('Color', 'w', 'Name', 'Speed_vs_Shield_Time');
99     plot(v_scan_list, results_score, 'b-', 'LineWidth', 1.5); hold on
100
101     scatter(best_v, max_score, 100, 'r', 'filled', 'p');
102
103     % 标记关键点
104     xlabel('飞行速度(m/s)', 'FontSize', 12);
105     ylabel('最大有效遮蔽时长(s)', 'FontSize', 12);
106     title('全速域战术性能分布图(PSO搜索)', 'FontSize', 14);
107     grid on;
108
109     % 自动寻找局部峰值 (比如高速区的峰值)
110     [pks, locs] = findpeaks(results_score);

```

```

105     if ~isempty(pks)
106         % 标记除了全局最优外的另一个显著峰值（如果有）
107         sorted_pks = sort(pks, 'descend');
108         if length(sorted_pks) > 1 && sorted_pks(2) > 4.0
109             second_pk = sorted_pks(2);
110             % 找到对应速度
111             idx_sec = find(results_score == second_pk, 1);
112             sec_v = v_scan_list(idx_sec);
113             if abs(sec_v - best_v) > 10 % 确保两个峰隔得够远
114                 plot(sec_v, second_pk, 'ko', 'MarkerFaceColor', 'y');
115                 text(sec_v, second_pk+0.1, sprintf('次优解\n%.1fm/s',
116                     sec_v), 'HorizontalAlignment', 'center');
116             end
117         end
118     end
119
120     text(best_v, max_score + 0.15, sprintf('全局最优\nV=%.1f\nT=%.4fs
121         ', best_v, max_score), ...
122         'Color', 'r', 'HorizontalAlignment', 'center', 'FontWeight',
123         'bold');
124
125     %% 5. 最终验证（使用 0.0001s 航天级精度复核冠军参数）
126     fprintf('\n>>>□正在使用□0.1ms□精度复核冠军方案...\n');
127     final_check_score = Tactical_Sim_Engine(best_v, best_p(1), best_p
128         (2), best_p(3), 0.0001, Env);
129     fprintf('>>>□最终复核得分:□%.6f□秒\n', -final_check_score); % 注
130     意引擎返回负值
131
132     % 绘制冠军轨迹
133     Plot_3D_Final(best_v, best_p, Env);
134 end
135
136 %% --- 内部微型 PSO 求解器 ---
137 function [best_pos, best_val] = Run_Micro_PSO(v, lb, ub, Env)
138     % 粒子群参数
139     n_part = 30;
140     n_iter = 50;
141     w = 0.6; c1 = 1.5; c2 = 1.5;
142     n_vars = 3;

```

```

140 % 初始化
141 pos = repmat(lb, n_part, 1) + rand(n_part, n_vars) .* repmat(ub-
    lb, n_part, 1);
142
143 % [种子注入] 针对不同速度段注入经验值，加速收敛
144 if v < 90
145     pos(1,:) = [176.88, 0.01, 2.5]; % 低速经验
146 else
147     pos(1,:) = [178.46, 0.01, 3.3]; % 高速经验
148 end
149
150 vel = zeros(n_part, n_vars);
151
152 pbest_pos = pos;
153 pbest_val = zeros(n_part, 1);
154
155 gbest_pos = zeros(1, n_vars);
156 gbest_val = -1e9; % 初始化为很小的值
157
158 % 评估初始种群
159 for i = 1:n_part
160     % 注意：Sim引擎返回的是负值(为了fminsearch)，这里我们取相反数
        变回正值方便比较
161     val = -Tactical_Sim_Engine(v, pos(i,1), pos(i,2), pos(i,3),
        Env.dt, Env);
162     pbest_val(i) = val;
163     if val > gbest_val
164         gbest_val = val;
165         gbest_pos = pos(i,:);
166     end
167 end
168
169 % 迭代
170 for t = 1:n_iter
171     for i = 1:n_part
172         r1 = rand(1, n_vars); r2 = rand(1, n_vars);
173         vel(i,:) = w*vel(i,:) + c1*r1.*(pbest_pos(i,:)-pos(i,:))
            + c2*r2.*(gbest_pos-pos(i,:));
174         pos(i,:) = pos(i,:) + vel(i,:);
175

```



```

176         % 边界限制
177         pos(i,:) = max(pos(i,:), lb);
178         pos(i,:) = min(pos(i,:), ub);
179
180         val = -Tactical_Sim_Engine(v, pos(i,1), pos(i,2), pos(i,3), Env.dt, Env);
181
182         if val > pbest_val(i)
183             pbest_val(i) = val;
184             pbest_pos(i,:) = pos(i,:);
185         end
186         if val > gbest_val
187             gbest_val = val;
188             gbest_pos = pos(i,:);
189         end
190     end
191 end
192
193 best_pos = gbest_pos;
194 best_val = gbest_val;
195 end
196
197 %% --- 仿真核函数 (Cost Function) ---
198 function score = Tactical_Sim_Engine(v, a, td, ty, dt, Env)
199     % 返回负值用于最小化, 正值代表遮蔽时长
200
201     % 几何解算
202     ang_rad = deg2rad(a);
203     Dir_Vec = [cos(ang_rad), sin(ang_rad), 0];
204     Vel_Vec = Dir_Vec * v;
205
206     P_Drop = Env.Pos_FY1_Init + Vel_Vec * td;
207     P_Pop = P_Drop + Vel_Vec * ty;
208     P_Pop(3) = P_Pop(3) - 0.5 * Env.g * ty^2;
209
210     % 剪枝: 地下
211     if P_Pop(3) < 0
212         score = 0; return;
213     end
214

```

```

215     t_pop = td + ty;
216     t_start = max(0, t_pop);
217     t_end = min(Env.Dist_Total_M1/Env.V_M1, t_pop + Env.
        Time_smoke_last);
218
219     if t_start >= t_end
220         score = 0; return;
221     end
222
223     % 矢量化计算
224     t_vec = t_start : dt : t_end;
225     if isempty(t_vec), score=0; return; end
226
227     d_m_vec = Env.V_M1 * t_vec;
228     P_M_mat = Env.Pos_M1_Init' + Env.Dir_M1' * d_m_vec;
229     P_Smk_mat = P_Pop' - [0;0;Env.V_smoke_sink] * (t_vec - t_pop);
230
231     V_LOS_mat = Env.Pos_TrueTarget' - P_M_mat;
232     W_mat = P_Smk_mat - P_M_mat;
233
234     c1 = sum(W_mat .* V_LOS_mat, 1);
235     c2 = sum(V_LOS_mat .* V_LOS_mat, 1);
236
237     b = c1 ./ c2;
238
239     % 垂足修正
240     Pb = P_M_mat + V_LOS_mat .* b;
241     idx_less = b < 0; if any(idx_less), Pb(:, idx_less) = P_M_mat(:,
        idx_less); end
242     idx_more = b > 1; if any(idx_more), Pb(:, idx_more) = repmat(Env.
        Pos_TrueTarget', 1, sum(idx_more)); end
243
244     dists_sq = sum((P_Smk_mat - Pb).^2, 1);
245
246     count = sum(dists_sq <= Env.R_smoke^2);
247     score = -(count * dt); % 返回负值
248 end
249
250 %% --- 绘图 ---
251 function Plot_3D_Final(v, params, Env)

```

```

252     a=params(1); td=params(2); ty=params(3);
253     ang_rad = deg2rad(a);
254     Vel_Vec = [cos(ang_rad), sin(ang_rad), 0] * v;
255     P_Drop = Env.Pos_FY1_Init + Vel_Vec * td;
256     P_Pop = P_Drop + Vel_Vec * ty;
257     P_Pop(3) = P_Pop(3) - 0.5 * Env.g * ty^2;
258
259     figure('Color','w'); hold on; grid on; axis equal; view([-20,
        25]);
260     plot3(Env.Pos_TrueTarget(1), Env.Pos_TrueTarget(2), Env.
        Pos_TrueTarget(3), 'go', 'MarkerSize', 10, 'MarkerFaceColor','
        g');
261     plot3(Env.Pos_FakeTarget(1), Env.Pos_FakeTarget(2), Env.
        Pos_FakeTarget(3), 'kp', 'MarkerSize', 10, 'MarkerFaceColor','
        k');
262     plot3([Env.Pos_FY1_Init(1) P_Drop(1)], [Env.Pos_FY1_Init(2) P_Drop
        (2)], [Env.Pos_FY1_Init(3) P_Drop(3)], 'b-');
263     plot3([P_Drop(1) P_Pop(1)], [P_Drop(2) P_Pop(2)], [P_Drop(3) P_Pop
        (3)], 'k:');
264     plot3(P_Pop(1), P_Pop(2), P_Pop(3), 'rh', 'MarkerFaceColor','r','
        MarkerSize',14);
265     line([Env.Pos_M1_Init(1) Env.Pos_TrueTarget(1)], [Env.Pos_M1_Init
        (2) Env.Pos_TrueTarget(2)], [Env.Pos_M1_Init(3) Env.
        Pos_TrueTarget(3)], 'Color', [0.7 0.7 0.7], 'LineStyle', '--')
        ;
266     title(['Global_Optimal: V=' num2str(v) ' m/s']);
267     legend('True','Fake','FY1','Bomb','Pop','LOS');
268 end

```

Listing 3: Problem 3

```

1  clc; clear; close all;
2
3  %% 1. 参数初始化
4  % 粒子群算法参数
5  n_particles = 200;           % 增加粒子数以覆盖更大的搜索空间(8维)
6  n_iterations = 300;          % 增加迭代次数
7  c1 = 2.0; c2 = 2.0;
8  w_max = 0.9; w_min = 0.4;
9
10 % 变量设计 (8维):

```

```

11 % x(1): 速度 v (70-140)
12 % x(2): 航向 theta (0-2pi)
13 % x(3): 第1枚投放时刻 t1 (0-60)
14 % x(4): 第1枚与第2枚的时间间隔 dt12 (1-30) -> 保证了间隔>=1s
15 % x(5): 第2枚与第3枚的时间间隔 dt23 (1-30) -> 保证了间隔>=1s
16 % x(6): 第1枚起爆延时 delay1 (0-15)
17 % x(7): 第2枚起爆延时 delay2 (0-15)
18 % x(8): 第3枚起爆延时 delay3 (0-15)
19
20 lb = [70, 0, 0, 1, 1, 0, 0, 0];
21 ub = [140, 2*pi, 60, 20, 20, 15, 15, 15];
22
23 % 速度限制
24 v_max = 0.15 * (ub - lb);
25 v_min = -v_max;
26
27 %% 2. 种群初始化
28 particles = repmat(lb, n_particles, 1) + rand(n_particles, 8) .* (ub
    - lb);
29 velocities = zeros(n_particles, 8);
30
31 pbest_pos = particles;
32 pbest_val = inf(n_particles, 1);
33 gbest_pos = zeros(1, 8);
34 gbest_val = inf;
35
36 stagnation_counter = 0; % 停滞计数器，用于判断是否陷入局部最优
37
38 %% 3. PSO 主循环
39 disp('开始多烟幕弹协同优化_(引入灾变机制防止局部最优)...');
40 tic;
41
42 for iter = 1:n_iterations
43     w = w_max - (w_max - w_min) * iter / n_iterations;
44
45     current_gbest_improved = false;
46
47     for i = 1:n_particles
48         current_x = particles(i, :);
49         current_x(1) = round(current_x(1)); % 速度取整

```

```

50
51     % 边界处理
52     current_x = max(current_x, lb);
53     current_x = min(current_x, ub);
54
55     % 计算适应度（核心修改：支持3枚弹）
56     fitness = calculate_fitness_multi(current_x);
57
58     % 更新个体最优
59     if fitness < pbest_val(i)
60         pbest_val(i) = fitness;
61         pbest_pos(i, :) = current_x;
62     end
63
64     % 更新全局最优
65     if fitness < gbest_val
66         gbest_val = fitness;
67         gbest_pos = current_x;
68         current_gbest_improved = true;
69     end
70 end
71
72 % --- 灾变机制（防止陷入局部最优） ---
73 if current_gbest_improved
74     stagnation_counter = 0;
75 else
76     stagnation_counter = stagnation_counter + 1;
77 end
78
79 % 如果连续20代没有进步，触发“小灾变”：重置部分粒子
80 if stagnation_counter > 20 && iter < n_iterations - 50
81     disp(['\t\t\t\t>>\t检测到局部最优停滞，触发灾变重置\t(Iter\t'
82         num2str(iter) '\t)']);
83     % 保留前 10% 的精英，重置其余 90%
84     n_keep = round(n_particles * 0.1);
85     [~, sorted_idx] = sort(pbest_val);
86     for k = n_keep+1 : n_particles
87         idx = sorted_idx(k);
88         particles(idx, :) = lb + rand(1, 8) .* (ub - lb);
89         velocities(idx, :) = zeros(1, 8);

```

```

89         pbest_val(idx) = inf; % 重置历史记忆
90     end
91     stagnation_counter = 0;
92 end
93
94 % 粒子更新公式
95 for i = 1:n_particles
96     r1 = rand(1, 8);
97     r2 = rand(1, 8);
98
99     velocities(i,:) = w * velocities(i,:) ...
100         + c1 * r1 .* (pbest_pos(i,:) - particles(i,:)) ...
101         + c2 * r2 .* (gbest_pos - particles(i,:));
102
103     velocities(i,:) = max(min(velocities(i,:), v_max), v_min);
104     particles(i,:) = particles(i,:) + velocities(i,:);
105
106     % 变异操作 (5%概率)
107     if rand < 0.05
108         dim = randi(8);
109         particles(i, dim) = lb(dim) + rand * (ub(dim) - lb(dim));
110     end
111     particles(i,:) = max(min(particles(i,:), ub), lb);
112 end
113
114 % 进度显示
115 if mod(iter, 10) == 0 || iter == 1
116     real_time = 0;
117     if gbest_val < -1000, real_time = -(gbest_val + 10000); end
118     fprintf('迭代 %d/%d, 当前最佳遮蔽时长: %.5f 秒 (停滞: %d)\n',
119         ...
120         iter, n_iterations, real_time, stagnation_counter);
121 end
122 toc;
123
124 %% 4. 结果解析与输出
125 best_params = gbest_pos;
126 best_params(1) = round(best_params(1));
127

```

```

128 % 解码时间参数
129 t_drop1 = best_params(3);
130 t_drop2 = t_drop1 + best_params(4);
131 t_drop3 = t_drop2 + best_params(5);
132 t_delay1 = best_params(6);
133 t_delay2 = best_params(7);
134 t_delay3 = best_params(8);
135
136 % 最终高精度验证
137 final_time = calculate_masking_pure_multi(best_params, 0.00001);
138
139 fprintf('\n-----最终优化结果(Problem_3)-----\n');
140 fprintf('最大有效遮蔽时长: %.5f 秒\n', final_time);
141 fprintf('\n策略详情: \n');
142 fprintf('无人机速度: %d m/s\n', best_params(1));
143 fprintf('无人机航向: %.4f rad (%.2f 度)\n', best_params(2), rad2deg(
    best_params(2)));
144 fprintf('\n[烟弹1] 投放时刻: %.4f s, 起爆延时: %.4f s, 起爆时刻: %.4f s\n', t_drop1, t_delay1, t_drop1+t_delay1);
145 fprintf('[烟弹2] 投放时刻: %.4f s, 起爆延时: %.4f s, 起爆时刻: %.4f s\n', t_drop2, t_delay2, t_drop2+t_delay2);
146 fprintf('[烟弹3] 投放时刻: %.4f s, 起爆延时: %.4f s, 起爆时刻: %.4f s\n', t_drop3, t_delay3, t_drop3+t_delay3);
147 fprintf('注: 投放间隔分别为 %.4f s 和 %.4f s (均满足 >=1s 约束)\n',
    best_params(4), best_params(5));
148
149
150 %% ===== 子函数定义区域 =====
151
152 function fitness = calculate_fitness_multi(x)
153     % 1. 解码所有参数
154     v = x(1); theta = x(2);
155     t_d1 = x(3);
156     t_d2 = t_d1 + x(4); % 确保间隔
157     t_d3 = t_d2 + x(5); % 确保间隔
158     del1 = x(6); del2 = x(7); del3 = x(8);
159
160     % 2. 计算3枚弹的运动学参数
161     [init_pos_m1, v_vec_m1, target_pos] = get_env_params(v, theta);

```

```

162
163     [exp1, pos1] = get_bomb_kinematics(v, theta, t_d1, del1);
164     [exp2, pos2] = get_bomb_kinematics(v, theta, t_d2, del2);
165     [exp3, pos3] = get_bomb_kinematics(v, theta, t_d3, del3);
166
167     % 物理约束：任何一枚在地下爆炸都给予惩罚
168     if pos1(3)<0 || pos2(3)<0 || pos3(3)<0
169         fitness = 1e6; return;
170     end
171
172     % 3. 确定仿真时间范围
173     smoke_dur = 20;
174     t_hit = norm(init_pos_m1) / 300;
175
176     % 整个遮蔽过程是3个区间的并集，我们取最早起爆到最晚结束
177     t_start_global = min([exp1, exp2, exp3]);
178     t_end_global = min(max([exp1, exp2, exp3]) + smoke_dur, t_hit);
179
180     if t_start_global >= t_end_global
181         fitness = 1e5; return;
182     end
183
184     % --- 智能扫描策略 ---
185
186     % A. 粗扫描 (0.1s)
187     dt_coarse = 0.1;
188     T_coarse = t_start_global : dt_coarse : t_end_global;
189     if isempty(T_coarse), fitness = 1e5; return; end
190
191     min_dist_global = inf;
192     potential_times = [];
193
194     % 预计算导弹位置
195     Pm_c = init_pos_m1 + v_vec_m1 .* T_coarse(:);
196     Pt = target_pos;
197
198     for k = 1:length(T_coarse)
199         t = T_coarse(k);
200         pm = Pm_c(k,:);
201         vec_line = Pt - pm;

```



```

202     norm_line = norm(vec_line);
203
204     dist_min_k = inf;
205
206     % 检查3个烟雾团
207     clouds = [exp1, pos1; exp2, pos2; exp3, pos3]; % 3x4 matrix
208     for m = 1:3
209         t_exp = clouds(m, 1);
210         p_exp = clouds(m, 2:4);
211
212         % 只有在烟雾存续期内才计算
213         if t >= t_exp && t <= t_exp + smoke_dur
214             ps = p_exp;
215             ps(3) = ps(3) - 3 * (t - t_exp); % 下沉
216
217             vec_point = ps - pm;
218             d = norm(cross(vec_point, vec_line)) / norm_line;
219             if d < dist_min_k, dist_min_k = d; end
220         end
221     end
222
223     if dist_min_k < min_dist_global, min_dist_global = dist_min_k
224         ; end
225     if dist_min_k < 18, potential_times = [potential_times; t];
226     end
227
228 % B. 精细计算
229 if isempty(potential_times)
230     fitness = min_dist_global;
231 else
232     dt_fine = 0.0001;
233     window = 0.06;
234     all_fine_times = [];
235
236     for t_center = potential_times'
237         t_sub = (max(t_start_global, t_center - window) : dt_fine
238             : min(t_end_global, t_center + window))';
239         t_sub = round(t_sub / dt_fine) * dt_fine; % 网格对齐
240         all_fine_times = [all_fine_times; t_sub];

```

```
239     end
240
241     if isempty(all_fine_times), fitness = min_dist_global; return
242         ; end
243
244     T_fine = unique(all_fine_times);
245
246     Pm_f = init_pos_m1 + v_vec_m1 .* T_fine;
247
248     mask_count = 0;
249
250     % 向量化比较麻烦，这里用循环处理这3个球
251     % 为了加速，把3个球的信息展开
252     exp_times = [exp1, exp2, exp3];
253     pos_starts = [pos1; pos2; pos3];
254
255     for j = 1:length(T_fine)
256         t = T_fine(j);
257         pm = Pm_f(j,:);
258         vec_line = Pt - pm;
259         norm_line = norm(vec_line);
260
261         is_masked = false;
262
263         % 遍历3个球
264         for m = 1:3
265             if t >= exp_times(m) && t <= exp_times(m) + smoke_dur
266                 ps = pos_starts(m, :);
267                 ps(3) = ps(3) - 3 * (t - exp_times(m));
268
269                 vec_point = ps - pm;
270                 % 快速判断距离平方
271                 cp = cross(vec_point, vec_line);
272                 d2 = sum(cp.^2) / (norm_line^2);
273
274                 if d2 <= 100 % 10^2
275                     is_masked = true;
276                     break; % 只要被任意一个遮住就算遮住
277                 end
278             end
279         end
280     end
281 end
```

```

278
279         if is_masked
280             mask_count = mask_count + 1;
281         end
282     end
283
284     fitness = -(mask_count * dt_fine) - 10000;
285 end
286 end
287
288 function mask_time = calculate_masking_pure_multi(x, dt)
289     % 解码
290     v = x(1); theta = x(2);
291     t_d1 = x(3); t_d2 = t_d1 + x(4); t_d3 = t_d2 + x(5);
292     del1 = x(6); del2 = x(7); del3 = x(8);
293
294     [init_pos_m1, v_vec_m1, target_pos] = get_env_params(v, theta);
295     [exp1, pos1] = get_bomb_kinematics(v, theta, t_d1, del1);
296     [exp2, pos2] = get_bomb_kinematics(v, theta, t_d2, del2);
297     [exp3, pos3] = get_bomb_kinematics(v, theta, t_d3, del3);
298
299     smoke_dur = 20;
300     t_hit = norm(init_pos_m1) / 300;
301     t_start = min([exp1, exp2, exp3]);
302     t_end = min(max([exp1, exp2, exp3]) + smoke_dur, t_hit);
303
304     if t_start >= t_end, mask_time=0; return; end
305
306     T = t_start : dt : t_end;
307     count = 0;
308
309     exp_times = [exp1, exp2, exp3];
310     pos_starts = [pos1; pos2; pos3];
311
312     for k = 1:length(T)
313         t = T(k);
314         pm = init_pos_m1 + v_vec_m1 * t;
315         vec_line = target_pos - pm;
316         norm_line = norm(vec_line);
317

```

```

318         masked = false;
319         for m = 1:3
320             if t >= exp_times(m) && t <= exp_times(m) + smoke_dur
321                 ps = pos_starts(m, :);
322                 ps(3) = ps(3) - 3 * (t - exp_times(m));
323
324                 vec_point = ps - pm;
325                 d = norm(cross(vec_point, vec_line)) / norm_line;
326                 if d <= 10
327                     masked = true; break;
328                 end
329             end
330         end
331         if masked, count = count + 1; end
332     end
333     mask_time = count * dt;
334 end
335
336 function [init_pos_m1, v_vec_m1, target_pos] = get_env_params(v,
    theta)
337     init_pos_m1 = [20000, 0, 2000];
338     fake_target = [0, 0, 0];
339     dir_m1 = (fake_target - init_pos_m1) / norm(fake_target -
        init_pos_m1);
340     v_vec_m1 = dir_m1 * 300;
341     target_pos = [0, 200, 0];
342 end
343
344 function [t_explode, explode_pos] = get_bomb_kinematics(v_mag, theta,
    t_drop, t_delay)
345     init_pos_uav = [17800, 0, 1800];
346     v_vec_uav = [v_mag * cos(theta), v_mag * sin(theta), 0];
347     drop_pos = init_pos_uav + v_vec_uav * t_drop;
348
349     g = 9.8;
350     explode_pos = drop_pos;
351     explode_pos(1) = drop_pos(1) + v_vec_uav(1) * t_delay;
352     explode_pos(2) = drop_pos(2) + v_vec_uav(2) * t_delay;
353     explode_pos(3) = drop_pos(3) - 0.5 * g * t_delay^2;
354

```

```

355     t_explode = t_drop + t_delay;
356 end

```

Listing 4: Problem 4

```

1 function Problem4_MultiUAV_Strict()
2     clc; close all;
3
4     %% 1. 全局环境参数 (Env)
5     Env.g = 9.8;
6     Env.Pos_True = [0, 200, 0];
7     Env.Pos_M1 = [20000, 0, 2000];
8     Env.V_M1 = 300;
9     Env.Vec_M = [0, 0, 0] - Env.Pos_M1;
10    Env.Dist_M1 = norm(Env.Vec_M);
11    Env.Dir_M1 = Env.Vec_M / Env.Dist_M1;
12    Env.V_sink = 3;
13    Env.R_smk = 10;
14    Env.T_last = 20;
15
16    Env.Pos_UAVs = [
17        17800, 0, 1800; % FY1
18        12000, 1400, 1400; % FY2
19        6000, -3000, 700 % FY3
20    ];
21
22    %% 2. 优化参数设置 (已修正速度约束)
23
24    % 估算基准航向
25    Base_Ang1 = rad2deg(atan2(200 - 0, 0 - 17800));
26    Base_Ang2 = rad2deg(atan2(200 - 1400, 0 - 12000));
27    Base_Ang3 = rad2deg(atan2(200 + 3000, 0 - 6000));
28
29    % === [修正点]: 将速度下界从 60 改为 70 ===
30    % 变量顺序: [V, Ang, T_drop, T_delay] * 3架
31    LB = [70, Base_Ang1-45, 0, 1, 70, Base_Ang2-45, 0, 1, 70,
32        Base_Ang3-45, 0, 1];
33
34    UB = [140, Base_Ang1+45, 15, 8, 140, Base_Ang2+45, 35, 8, 140,
35        Base_Ang3+45, 55, 8];
36
37    de_opts.NP = 200;

```

```

35     de_opts.MaxIter = 800;
36     de_opts.F = 0.5;
37     de_opts.CR = 0.9;
38
39     fprintf('=====\\n
        ');
40     fprintf('□□□□□□问题4：三机协同立体封锁□(速度修正版□70-140)\\n');
41     fprintf('=====\\n
        ');
42
43     try, if isempty(gcp('nocreate')), parpool; end; end
44
45     CostFunc = @(x) CostFunc_3UAV(x, Env);
46
47     tic;
48     [best_x, best_val, curve] = Run_DE_3UAV(CostFunc, LB, UB, de_opts
        );
49     total_time = toc;
50
51     %% 3. 结果解析与输出
52     [~, real_shield_time] = CostFunc_3UAV(best_x, Env);
53     Res = Parse_Params(best_x);
54
55     fprintf('\\n>>>□优化完成！耗时：□%.2f□秒\\n', total_time);
56     fprintf('>>>□□最终最大遮蔽时长：□%.4f□秒\\n', real_shield_time);
57
58     fprintf('\\n[战术指令表]\\n');
59     fprintf('□□机号□□□□初始位置(X,Y,Z)□□□速度(m/s)□□□航向(deg)□□□投放(
        s)□□□延时(s)□□□起爆(s)□□\\n');
60     fprintf('
        |-----|-----|-----|-----|-----|-----|-----|
        n');
61     fprintf('□□FY1□□□□(□5d,□5d,□4d)□□□□9.2f□□□9.2f□□□7.2f□□□7.2f□
        □□7.2f□\\n', ...
        Env.Pos_UAVs(1,:), Res(1).v, Res(1).a, Res(1).td, Res(1).ty,
        Res(1).tp);
62     fprintf('□□FY2□□□□(□5d,□5d,□4d)□□□□9.2f□□□9.2f□□□7.2f□□□7.2f□
        □□7.2f□\\n', ...
        Env.Pos_UAVs(2,:), Res(2).v, Res(2).a, Res(2).td, Res(2).ty,
        Res(2).tp);

```

```

65     fprintf(' |FY3| |(%5d,%5d,%4d) | |%9.2f| |%9.2f| |%7.2f| |%7.2f| |%7.2f| \n', ...
66         Env.Pos_UAVs(3,:), Res(3).v, Res(3).a, Res(3).td, Res(3).ty,
67         Res(3).tp);
68
69     %% 4. 绘图
70     Plot_3UAV_Scenario(Res, real_shield_time, curve, Env);
71     Save_To_Excel(Res, 'result2.xlsx');
72
73     %% =====
74     % 局部函数库（保持不变，仅 CostFunc_3UAV 内部也加一道保险）
75     % =====
76
77     function Res = Parse_Params(x)
78         for i = 1:3
79             idx = (i-1)*4;
80             Res(i).v = x(idx+1);
81             Res(i).a = x(idx+2);
82             Res(i).td = x(idx+3);
83             Res(i).ty = x(idx+4);
84             Res(i).tp = Res(i).td + Res(i).ty;
85         end
86     end
87
88     function [score, pure_time] = CostFunc_3UAV(x, Env)
89         Res = Parse_Params(x);
90
91         % --- [保险修正]：并在内部逻辑中再次检查边界 ---
92         for i=1:3
93             if Res(i).v < 70 || Res(i).v > 140 % 严格限制 70-140
94                 score = 1e6; pure_time = 0; return;
95             end
96         end
97
98         % ...（其余物理计算逻辑与之前完全一致，此处省略以节省篇幅）...
99         % 您之前复制的 CostFunc_3UAV 代码逻辑在这里保持不变
100
101         % ----- 粘贴之前的 CostFunc_3UAV 逻辑 -----
102         P_Pop = zeros(3, 3);

```

```

103     Times_Pop = zeros(3, 1);
104     for i = 1:3
105         ang_rad = deg2rad(Res(i).a);
106         Vel_Vec = [cos(ang_rad), sin(ang_rad), 0] * Res(i).v;
107         P_Drop = Env.Pos_UAVs(i,:) + Vel_Vec * Res(i).td;
108         Disp = Vel_Vec * Res(i).ty;
109         Disp(3) = Disp(3) - 0.5 * 9.8 * Res(i).ty^2;
110         P_Pop(i,:) = P_Drop + Disp;
111         Times_Pop(i) = Res(i).tp;
112         if P_Pop(i,3) < 0, score = 1e5; pure_time = 0; return; end
113     end
114
115     t_start = min(Times_Pop);
116     t_end = min(Env.Dist_M1/Env.V_M1, max(Times_Pop) + Env.T_last);
117     if t_start >= t_end, score = 1e5; pure_time = 0; return; end
118
119     Target_Points = [0, 200, 5; 0, 200, 10; 0, 200, 0; -7, 200, 5; 7,
120                     200, 5];
121     num_pts = 5;
122     dt = 0.05; t_vec = t_start : dt : t_end;
123     total_coverage = 0;
124     penalty_dist = 0;
125     min_dists = [1e9, 1e9, 1e9];
126
127     for t = t_vec
128         P_M = Env.Pos_M1 + Env.Dir_M1 * (Env.V_M1 * t);
129         blocked_pts_count = 0;
130         for p = 1:num_pts
131             TP = Target_Points(p,:);
132             v_los = TP - P_M;
133             len_los_sq = sum(v_los.^2);
134             is_pt_blocked = false;
135             for k = 1:3
136                 if t >= Times_Pop(k) && t <= Times_Pop(k) + Env.
137                     T_last
138                     P_Smk = P_Pop(k,:) - [0, 0, Env.V_sink * (t -
139                         Times_Pop(k))];
140                     w_vec = P_Smk - P_M;
141                     c1 = dot(w_vec, v_los);
142                     if c1 > 0

```



```

140         b = max(0, min(1, c1 / len_los_sq));
141         Pb = P_M + v_los * b;
142         d_sq = sum((P_Smk - Pb).^2);
143         dist = sqrt(d_sq);
144         if p == 1 && dist < min_dists(k), min_dists(k)
            = dist; end
145         if d_sq <= Env.R_smk^2, is_pt_blocked = true;
            end
146         end
147     end
148 end
149     if is_pt_blocked, blocked_pts_count = blocked_pts_count +
        1; end
150 end
151     total_coverage = total_coverage + (blocked_pts_count /
        num_pts) * dt;
152 end
153 pure_time = total_coverage;
154 for k=1:3, penalty_dist = penalty_dist + max(0, min_dists(k) -
        Env.R_smk); end
155 score = -total_coverage + 0.1 * penalty_dist;
156 end
157
158 %% --- DE 求解器（修正种子速度）---
159 function [best_mem, best_val, curve] = Run_DE_3UAV(cost_func, lb, ub,
    opts)
160     NP = opts.NP; D = length(lb);
161     pop = repmat(lb, NP, 1) + rand(NP, D) .* repmat(ub-lb, NP, 1);
162
163     % === [修正点]：种子的速度也必须符合 70-140 约束 ===
164     % FY1 修正为 73（低速流最优），FY2/FY3 为 120（高速流）
165     pop(1,:) = [73, 180, 0, 3, 120, 190, 15, 4, 120, 150, 35, 5];
166     pop(2,:) = [140, 180, 0, 3, 140, 190, 20, 3, 140, 150, 40, 3];
167
168     val = zeros(NP, 1);
169     parfor i=1:NP, val(i) = cost_func(pop(i,:)); end
170     [best_val, idx] = min(val);
171     best_mem = pop(idx, :);
172     curve = zeros(opts.MaxIter, 1);
173

```

```

174     h = waitbar(0, 'Multi-UAV_Optimization...');
175     for gen = 1 : opts.MaxIter
176         F = opts.F * (1 - 0.3 * gen/opts.MaxIter);
177         pop_new = pop; val_new = val;
178         parfor i = 1 : NP
179             r = randperm(NP, 3);
180             mutant = best_mem + F * (pop(r(1),:) - pop(r(2),:));
181             trial = pop(i, :);
182             j_rand = randi(D);
183             for j = 1 : D
184                 if rand < opts.CR || j == j_rand, trial(j) = mutant(j); end
185             end
186             trial = max(trial, lb); trial = min(trial, ub);
187             t_v = feval(cost_func, trial);
188             if t_v < val(i), pop_new(i,:) = trial; val_new(i) = t_v; end
189         end
190         pop = pop_new; val = val_new;
191         [c_best, idx] = min(val);
192         if c_best < best_val, best_val = c_best; best_mem = pop(idx, :); end
193         curve(gen) = -best_val;
194         if mod(gen, 50) == 0
195             [~, t_real] = feval(cost_func, best_mem);
196             waitbar(gen/opts.MaxIter, h, sprintf('Iter_%d: %.2fs', gen, t_real));
197             fprintf('Iter_%d: 引导得分 %.4f | 真实遮蔽 %.4fs\n', gen, best_val, t_real);
198         end
199     end
200     close(h);
201 end
202
203 %% --- 绘图与保存（保持不变） ---
204 function Plot_3UAV_Scenario(Res, score, curve, Env)
205     figure('Color','w', 'Position', [50, 50, 1200, 600]);
206     subplot(1, 2, 1); hold on; grid on; axis equal; view([-30, 30]);
207     plot3(Env.Pos_True(1), Env.Pos_True(2), Env.Pos_True(3), 'go', 'MarkerSize', 12, 'MarkerFaceColor','g');

```

```

208     plot3(Env.Pos_M1(1), Env.Pos_M1(2), Env.Pos_M1(3), 'rv', '
        MarkerSize', 10, 'MarkerFaceColor','r');
209     colors = {'b', 'm', [0 0.5 0]};
210     for i = 1:3
211         ang = deg2rad(Res(i).a);
212         V = [cos(ang), sin(ang), 0] * Res(i).v;
213         P0 = Env.Pos_UAVs(i,:);
214         P_Drop = P0 + V * Res(i).td;
215         plot3([P0(1) P_Drop(1)], [P0(2) P_Drop(2)], [P0(3) P_Drop(3)
            ], '-', 'Color', colors{i}, 'LineWidth', 1.5);
216         text(P0(1), P0(2), P0(3)+200, sprintf('FY%d',i), 'Color',
            colors{i});
217         P_Pop = P_Drop + V * Res(i).ty - [0,0,0.5*9.8*Res(i).ty^2];
218         plot3(P_Pop(1), P_Pop(2), P_Pop(3), 'h', 'MarkerSize', 14, '
            MarkerFaceColor', colors{i});
219         plot3([P_Drop(1) P_Pop(1)], [P_Drop(2) P_Pop(2)], [P_Drop(3)
            P_Pop(3)], 'k:');
220     end
221     line([Env.Pos_M1(1) Env.Pos_True(1)], [Env.Pos_M1(2) Env.Pos_True
        (2)], [Env.Pos_M1(3) Env.Pos_True(3)], ...
222         'Color', [0.6 0.6 0.6], 'LineStyle', '--');
223     xlabel('X'); ylabel('Y'); zlabel('Z');
224     title('三机协同立体封锁示意图');
225     subplot(1, 2, 2); hold on; grid on;
226     for i=1:3
227         tp = Res(i).tp;
228         fill([tp, tp+Env.T_last, tp+Env.T_last, tp], [i-0.3, i-0.3, i
            +0.3, i+0.3], colors{i}, 'FaceAlpha', 0.6);
229         text(tp, i, sprintf('□Start:□%.1fs', tp));
230     end
231     title(['协同遮蔽时序□(总长:□' num2str(score) 's)']);
232     xlabel('时间□(s)'); ylim([0 4]); yticks([1 2 3]); yticklabels({'
        FY1','FY2','FY3'});
233 end
234
235 function Save_To_Excel(Res, filename)
236     data = zeros(3, 5);
237     for i=1:3
238         data(i,1) = Res(i).v;
239         data(i,2) = Res(i).a;

```

```
240         data(i,3) = Res(i).td;
241         data(i,4) = Res(i).ty;
242         data(i,5) = Res(i).tp;
243     end
244     try, writematrix(data, filename); catch, end
245 end
```