烟幕弹投放策略的多重优化模型与求解

摘要

本文围绕烟幕干扰弹的多无人机协同投放策略优化问题，基于运动学分析、几何光学原理与智能优化算法，系统性地构建了适用于不同任务场景的烟幕遮蔽模型与决策框架。针对五项子问题，分别提出了具有针对性的建模方法与求解策略，并得到合理的投放策略。具体研究内容如下：

针对问题一：我们基于多体运动学建立了烟幕弹、下沉云团及来袭导弹的运动轨迹模型，结合光学制导机理与凸集凸锥理论，提出了一种高效的空间遮蔽判定准则。通过模型耦合与时间区间变步长搜索策略，精确计算出对 M1 导弹的有效遮蔽时长为 **1.410s**，其主要遮蔽窗口为 **[7.970s, 9.380s]**。

针对问题二：我们构建了以遮蔽时间最大化为目标的单目标优化模型，决策变量包括无人机飞行速度、航向角、烟幕弹投放时间与起爆延迟。在继承问题一模型基础上，引入**粒子群优化算法（PSO）与自适应步长搜索机制**进行参数协同优化，最终获得最大遮蔽时间 **4.59s**。在模型检验阶段，我们发现改进后的 PSO 算法迭代约 50 次后收敛稳定，充分证明了该模型与优化结果的稳健性和可靠性。

针对问题三：我们针对单机连续投放多枚烟幕弹的复杂场景，建立了包含 8 个决策变量的系统优化模型，采用改进**差分进化算法（DE）**进行求解，并引入惩罚函数机制以确保每枚烟幕弹均发挥有效遮蔽作用。最终得到总遮蔽时长为 **6.75s**，三枚烟幕弹的遮蔽区间分别为 **[0.61s,4.11s]**、**[3.51s,6.51s] 和 [6.35s,7.35s]**。多次随机初始化的重复实验表明，算法具有较高的稳定性和鲁棒性。

针对问题四：我们研究了三架无人机协同干扰场景，构建了包含 12 个决策变量的高维优化模型。采用**遗传算法（GA）与模拟退火（SA）相结合的混合优化策略**，在保障全局搜索能力的同时提升局部寻优精度。最终实现总遮蔽时间 **11.42s**，三机遮蔽时段分别为 **[0.88s,5.48s]**、**[17.51s,21.33s] 和 [26.64s,29.65s]，且互不重叠**，体现出良好的时序协同性与任务分配合理性。

针对问题五：为了因对五架无人机对三枚导弹的协同干扰场景，我们建立了包含 40 个决策变量的复杂约束优化模型，重点引入了多导弹遮蔽判别逻辑和无人机间碰撞规避约束。采用**改进的 GA-SA 混合算法**进行求解，最终综合遮蔽时间（并集）达到 **22.98** 秒。通过参数灵敏度分析、三维轨迹仿真和冲突检测验证，表明该策略在复杂环境下仍具备良好的可行性和实战潜力。

关键字：烟幕干扰　PSO　差分进化算法　遗传算法　模拟退火算法

一、 问题重述

1.1 问题背景

烟幕干扰弹作为一种典型的无源干扰弹药，凭借其低成本、高效费比的技术优势，在现代军事对抗领域中获得了广泛的应用。该类弹药主要依托化学燃烧反应或爆炸驱动机制，使装填的发烟剂分散形成烟幕或气溶胶遮蔽云团，通过在被保护目标前方特定空域构建光学、红外或雷达波段的遮蔽屏障，实现对敌方制导导弹的有效干扰，进而达成对特定作战目标的防护效能。

随着烟幕干扰技术的持续迭代与发展，当前已形成多种成熟的投放与控制技术体系。通过采用精确制导投放手段，可在烟幕干扰弹的发烟剂抛撒前，实现其在预定空间位置的精准抵达；同时，借助时间引信的时序精确控制技术，能够对弹药的起爆时刻进行调控，从而构建起烟幕干扰弹的定点精确抛撒能力。在此背景下，如何进一步提升烟幕干扰弹的抛撒精度，已成为当前烟幕干扰技术领域亟待深入研究的关键问题之一。

1.2 问题要求

基于上述背景，要求建立数学模型解决一下问题。

现有真、假两个目标，真目标为一个下底面圆心位于 (0,200,0) 的圆柱体，假目标位于原点。考虑利用无人机实现烟雾干扰弹的投放，用以保护真目标。已知来袭空地导弹共有三枚，飞行速度 300m/s，方向直指假目标。警戒雷达已经发现了 3 枚导弹的具体位置。

区域上空有五架无人机，位置信息给定。无人机受领任务后可瞬时调整飞行方向，再以 70-140m/s 速度等高度匀速直线飞行。每架无人机投放两枚烟幕干扰弹至少间隔 1s。已知烟幕干扰弹脱离无人机后受重力运动，起爆后瞬时形成球状烟幕云团，并以 3m/s 匀速下沉，云团中心 10m 范围内、起爆 20s 内可提供有效遮蔽。具体问题如下：

1 用 FY1 投放 1 枚烟幕弹干扰 M1，FY1 以 120m/s 朝假目标飞行，受领任务 1.5s 后投放，间隔 3.6s 起爆，求有效遮蔽时长。
2 用 FY1 投放 1 枚烟幕弹干扰 M1，确定 FY1 飞行方向、速度及烟幕弹投放点、起爆点，使遮蔽时间最长。
3 用 FY1 投放 3 枚烟幕弹干扰 M1，给出投放策略
4 用 FY1、FY2、FY3 各投放 1 枚烟幕弹干扰 M1，给出投放策略
5 用 5 架无人机（每架至多投 3 枚）干扰 M1、M2、M3，给出投放策略

二、 问题分析

2.1 问题一分析

问题一是在单个无人机发射单枚烟幕干扰弹时，计算单一导弹的有效遮蔽时长，其核心在于构建多目标运动模型与遮蔽判定准则。依据给定的无人机、烟雾弹以及云团的运动状态参数，我们可以分别建立这三个目标各自的运动模型。其次，我们需要根据烟雾弹的工作原理，给出烟幕弹有效遮挡导弹的判定准则，以此作为判断某时刻遮蔽是否成立的标准。最终的计算可以通过选择合理步长，遍历云团有效存在时间段，结合运动模型和遮蔽判定模型，统计出总的有效遮蔽时长。

2.2 问题二分析

问题二仍然是在处理单个无人机发射单枚烟幕干扰弹干扰单枚导弹的情形，但是此时并未给出无人机飞行姿态等各项参数，而是需要我们选择最优的参数组合以最大化有效遮蔽时长。我们可以基于第一问已经构建的各类模型，通过选择合适的优化算法，搜索出最优参数组合，从而提供最优投放策略。具体求解过程可以使用粒子群优化算法。

2.3 问题三分析

问题三在问题一“单一遮蔽时长计算”与问题二“单机单弹参数优化”的基础上，进一步拓展场景复杂度，聚焦单架无人机投放三枚烟幕弹干扰单枚导弹的场景，核心目标是在保证每枚烟幕弹均能有效遮蔽真目标的前提下，通过多维度参数优化，最大化总有效遮蔽时长。我们具体采用差分进化算法更加适配该问题的求解。

2.4 问题四分析

问题四在前三问基础上，进一步升级为多机协同干扰，聚焦三架独立无人机各投放一枚烟幕弹，核心目标是通过 12 个高维决策变量的协同优化，在保证每枚烟幕弹均有效遮蔽真目标的前提下，最大化总有效遮蔽时长。我们基于 GA-SA 混合算法求解，可得到三架无人机的最优参数组合与总有效遮蔽时长。

2.5 问题五分析

问题五要求通过优化五架无人机的飞行速度、航向角及多枚烟幕弹的投放与起爆时序，在严格满足时空约束与避碰条件下，最大化三枚导弹的有效遮蔽时间。问题复杂度进一步上升。通过建立多变量非线性优化模型，并采用智能算法求解，在严格满足时空与避碰约束条件下，实现了多导弹协同遮蔽效果的最大化。

三、 模型假设

为了简化模型，并排除次要因素的影响，我们给出如下四个模型假设：

- **忽略假目标和导弹形状**：由于此题物体的运动区域相当大，因此假目标与导弹的形状对本题影响忽略不计，可视作质点。
- **忽略云团的扩散和形状**：由于烟雾弹爆炸形成的云团在半径 10m 内才具有遮挡效果，故云团扩散部分对本题研究对象没有影响，从而可以把云团视为一个规则球体。
- **简化光线传播**：只考虑光沿直线传播，不受其他因素影响。
- **排除影响各种物体运动的次要因素**：在分析物体运动时，只考虑动力学的影响，不考虑其它环境因素 (如空气阻力等)。
- **忽略信号传播时间**：由于电信号的传播极快，控制雷达向无人机传达飞行任务耗时相当小，可以忽略。
- **简化导弹类型**：我们只考虑导弹为光学制导导弹，即一旦导弹和导弹所要攻击的目标之间的视线全部被遮挡，那么导弹被成功干扰。

四、 符号说明

表 1

符号	含义
v_{FY1}	无人机 FY1 的飞行速度，单位为米每秒（m/s）
t_{drop}	烟幕弹投放时刻，单位为秒（s）
t_{det}	烟幕弹起爆时刻，单位为秒（s）
v_{M1}	导弹 M1 的飞行速度，单位为米每秒（m/s）
d	导弹到烟幕球心的距离，即 $\lvert\overrightarrow{PO}\rvert$
$\theta_{\text{smoke}}(t)$	烟幕遮蔽锥体的半顶角，单位为弧度（rad）
v	无人机飞行速度（优化变量），单位为米每秒（m/s）
ϕ	无人机飞行方位角（优化变量），单位为弧度（rad）
t_{launch}	烟幕弹投放时间（优化变量），单位为秒（s）
t_{delay}	烟幕弹起爆延迟时间（优化变量），单位为秒（s）
t_{detonate}	烟幕弹起爆时间，单位为秒（s）
$T_{\text{effective}}$	单枚烟幕弹的有效遮蔽时间，单位为秒（s）
t_{launch1}	首次烟幕弹投放时间（优化变量），单位为秒（s）
$t_{\text{delay}k}$	第 k 枚烟幕弹的起爆延迟时间（优化变量），单位为秒（s）
T_{total}	多枚烟幕弹的总有效遮蔽时间，单位为秒（s）
$t_{\text{launch},i}$	第 i 架无人机的烟幕弹投放时间（优化变量），单位为秒（s）
$t_{\text{delay},i}$	第 i 架无人机的烟幕弹起爆延迟时间（优化变量），单位为秒（s）
$t_{\text{detonate},i}$	第 i 架无人机的烟幕弹起爆时间，单位为秒（s）
S_{ijk}	无人机 i 第 k 枚烟幕弹对导弹 i 的有效遮蔽时间区间
T_{Mi}	导弹 i 的飞行总时间，单位为秒（s）

五、 问题一模型

5.1 模型的准备

问题一中最核心的问题就是如何确定烟幕干扰弹实现了有效的遮蔽效果。在遮蔽判断模型的构建过程中，我们利用了“凸集凸锥关系定理”：对于对应真目标的凸集 C 和对应导弹视线锥体的凸锥 K 而言，C 的上下表面齐次坐标点均含于 K，与 C 的齐次坐标点经非负实数缩放后形成的射线簇均含于 K，这两个说法等价。这个定理成功把“烟幕是否完全遮蔽真目标”的几何判定问题转化为“凸集与凸锥的包含关系问题”，从而大大提高了判断遮蔽效果的效率和准确性。

我们首先给出凸集和凸锥的定义：

定义 1 (凸锥) 集合 $K \subseteq \mathbb{R}^n$ 称为凸锥，若对任意 $x, y \in K$ 和任意 $\lambda, \mu \geq 0$，有 $\lambda x + \mu y \in K$。

定义 2 (凸集) 集合 $C \subseteq \mathbb{R}^n$ 称为凸集，若对任意 $x, y \in C$ 和任意 $\theta \in [0, 1]$，有 $\theta x + (1-\theta)y \in C$。

下面是“凸集和凸锥的关系定理”的叙述和严格证明

定理 1 设 $C \subseteq \mathbb{R}^n$ 为非空凸集，$K \subseteq \mathbb{R}^{n+1}$ 为凸锥。定义：

$$
\begin{aligned}
C_{top} &= \{(x, 1) \in \mathbb{R}^{n+1} \mid x \in C\} \quad \text{(顶面)}, \\
C_{bottom} &= \{(x, 0) \in \mathbb{R}^{n+1} \mid x \in C\} \quad \text{(底面)}, \\
\overline{C} &= \{\lambda(x, 1) \mid x \in C, \lambda \geq 0\} \quad \text{(射线簇)}.
\end{aligned}
$$

则以下两命题等价：

1. $C_{top} \subseteq K$ 且 $C_{bottom} \subseteq K$；
2. $\overline{C} \subseteq K$。

证明 1 一个方面，设 $C_{top} \subseteq K$ 且 $C_{bottom} \subseteq K$，需证 $\forall x \in C, \lambda \geq 0$，有 $\lambda(x, 1) \in K$。下面分两种情况讨论 λ:

- 若 $\lambda = 0$：则 $\lambda(x, 1) = (0, 0)$。由 $C_{bottom} \subseteq K$ 取 $x \in C$，得 $(x, 0) \in K$；又 K 是锥，取 $\alpha = 0$，故 $0 \cdot (x, 0) = (0, 0) \in K$。
- 若 $\lambda > 0$：由 $C_{top} \subseteq K$ 得 $(x, 1) \in K$；又 K 是锥，故 $\lambda(x, 1) \in K$

综上，$\overline{C} \subseteq K$。

另一方面，设 $\overline{C} \subseteq K$，需证 $C_{top} \subseteq K$ 且 $C_{bottom} \subseteq K$。*1.* 证 $C_{top} \subseteq K$：取 $\lambda = 1$，则 $\forall x \in C$，有 $1 \cdot (x, 1) = (x, 1) \in \overline{C} \subseteq K$，故 $C_{top} \subseteq K$。*2.* 证 $C_{bottom} \subseteq K$：取序列 $\lambda_k = \frac{1}{k}$ $(k \in \mathbb{N}^+)$，则 $\forall x \in C$，有 $\lambda_k(x, 1) = \left(\frac{x}{k}, \frac{1}{k}\right) \in \overline{C} \subseteq K$。

由于 K 是闭凸锥（凸锥闭性保证极限封闭），令 $k \to \infty$，则 $\left(\frac{x}{k}, \frac{1}{k}\right) \to (0,0) \in K$。进一步，对任意 $\varepsilon > 0$，由 K 凸性：

$$(1-\varepsilon)(0,0) + \varepsilon(x,1) = (\varepsilon x, \varepsilon) \in K \square$$

再由 K 锥性，令 $\varepsilon \to 0$，其极限 $(x,0) \in K$（闭集对极限封闭）。故 $C_{bottom} \subseteq K$。

综上，两命题等价。

5.2 模型的建立

对于单个无人机发射单枚烟幕干扰弹的运动模型，我们不妨基于笛卡尔坐标系进行考虑。

以假目标为原点 $O(0,0,0)$，水平面为 xy 平面，竖直方向为 z 轴，构建笛卡尔坐标系。真目标为圆柱体，其下底面圆心坐标为 $(0,200,0)$，半径 $r = 7\,\mathrm{m}$，高度 $h = 10\,\mathrm{m}$，因此真目标圆柱体的空间范围可表示为：

$$\begin{cases} x^2 + (y-200)^2 \le 7^2 \\ 0 \le z \le 10 \end{cases}$$

现在考虑物体运动模型以及烟雾遮挡判断模型。

5.2.1 物体运动模型

该模型包括三个子模型，分别对应三个物体各自的运动模型。

无人机 FY1 的运动模型

已知条件如下：初始时刻（即警戒雷达探测到导弹时刻）无人机 FY1 的位置为：

$$P_{\mathrm{FY1},0} = (17800, 0, 1800)\mathrm{m}$$

其飞行速度为恒定值：

$$v_{\mathrm{FY1}} = 120\mathrm{m/s}$$

飞行方向指向假目标位置（坐标原点），相应的单位方向向量为：

$$\vec{u}_{\mathrm{FY1}} \approx (-0.9948, 0, -0.0995)$$

无人机在接收到任务指令后，需经过 1.5s 的延迟方可实施烟幕弹投放，故投放时刻为：

$$t_{\mathrm{drop}} = 1.5\mathrm{s}$$

其中时间 t 以雷达发现导弹的时刻为零点（$t = 0$）。

据此，无人机在任意时刻 $t \in [0, 1.5]$ 内的位置可表示为时间 t 的函数：

$$P_{\mathrm{FY1}}(t) = P_{\mathrm{FY1},0} + v_{\mathrm{FY1}} \cdot \vec{u}_{\mathrm{FY1}} \cdot t$$

烟幕弹投放点的坐标即为 $t = 1.5$s 时无人机的位置：

$$\begin{aligned} P_{\text{drop}} &= P_{\text{FY1}}(1.5) \\ &\approx (17501.52, 0, 1782.09)\text{m} \end{aligned}$$

烟幕云团的运动模型

已知条件如下：烟幕弹在投放后延迟 3.6s 起爆，故起爆时刻为：

$$t_{\text{det}} = t_{\text{drop}} + 3.6 = 5.1\text{s}$$

起爆后，烟幕云团呈球体形态，其中心以 3m/s 的速度沿 z 轴负方向匀速下沉，半径为 $R = 10$m。该模型在起爆后 20s 内有效，即有效时段为 $t \in [5.1, 25.1]$s。起爆时刻的云团球心与投放点位置重合，即：

$$O_{\text{smoke},0} = P_{\text{drop}} \approx (17501.52, 0, 1782.09)\text{m}$$

据此，烟幕云团球心在任意时刻 $t \in [5.1, 25.1]$ 中的位置可表示为：

$$O_{\text{smoke}}(t) = (17501.52,\ 0,\ 1782.09 - 3 \cdot (t - 5.1))$$

烟幕云团所覆盖的空间范围定义为所有满足以下不等式的点 (x, y, z) 的集合：

$$\sqrt{(x - 17501.52)^2 + y^2 + (z - (1782.09 - 3(t - 5.1)))^2} \leq 10$$

导弹 M1 的运动模型

已知条件如下：初始时刻（$t = 0$）导弹 M1 的空间位置为：

$$P_{\text{M1},0} = (20000, 0, 2000)\ \text{m}$$

其飞行速度为恒定值：

$$v_{\text{M1}} = 300\ \text{m/s}$$

飞行方向指向假目标位置（坐标原点），相应的单位方向向量可通过归一化位置向量得到：

$$\vec{u}_{\text{M1}} \approx (-0.9950, 0, -0.0995)$$

据此，导弹在任意时刻 $t \geq 0$ 的空间位置可表示为时间 t 的函数：

$$P_{\text{M1}}(t) = P_{\text{M1},0} + v_{\text{M1}} \cdot \vec{u}_{\text{M1}} \cdot t$$

展开为分量形式：

$$P_{\text{M1}}(t) = (20000 - 300 \cdot 0.9950 \cdot t,\ 0,\ 2000 - 300 \cdot 0.0995 \cdot t)$$

5.2.2 烟雾有效遮挡的判断模型

模型依据和思路

我们已经假设研究的导弹都属于光学制导导弹，故烟雾弹产生的云团（半径为 10m 的圆球）必须遮挡住圆柱体（真目标）表面所有点和导弹（质点）的连线，在几何上**即为弹 M1 到真目标任意点的视线（射线）均需穿过烟幕云团（球体）**，而这又等价于**圆柱体完全按包含于导弹（视为质点）和云团形成的“遮挡圆锥”里面**，其中遮挡圆锥的顶点为导弹，母线为从导弹出发并与云团（圆球）相切的射线（(见图 1)）。

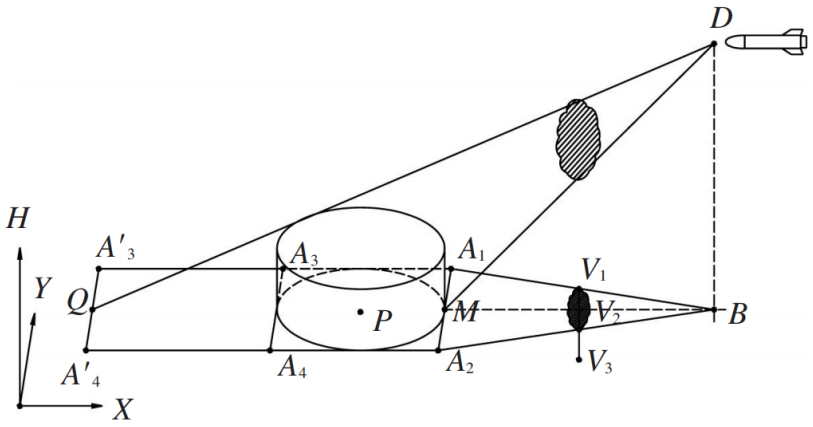

图1　被保护目标幅员等效平面示意图

图 1　遮挡判断图

又由凸集和凸锥的关系定理的可知，若凸集 C 的顶面 C_{top} 和底面 C_{bottom} 均包含于凸锥 K，则 C 生成的射线簇 $\overline{C} = \{\lambda(x,1) \mid x \in C, \lambda \geq 0\}$ 也包含于 K，故我们只需判断**圆柱体（真目标）的上下表面包含于导弹烟幕云团与导弹形成的凸锥**即可。这大大简化了计算的时间复杂度，在后面解决更复杂的多变量高维问题时发挥着重要作用。

模型的具体形式

设导弹位置为 $P = P_{\text{M1}}(t)$，烟幕云团球心为 $O = O_{\text{smoke}}(t)$，真目标上任意点为 Y（满足 $x^2 + (y-200)^2 \leq 7^2$ 且 $0 \leq z \leq 10$）。视线向量为 $\overrightarrow{PY} = Y - P$，导弹到烟幕球心的向量为 $\overrightarrow{PO} = O - P$，记 $|\overrightarrow{PO}| = d$。烟幕锥体以 P 为顶点，$\overrightarrow{PO}$ 为轴线方向，半顶角 $\theta_{\text{smoke}}(t)$ 满足：

$$\cos\theta_{\text{smoke}}(t) = \frac{\sqrt{d^2 - 10^2}}{d} \tag{1}$$

视线 $\overrightarrow{PY}$ 与烟幕锥体轴线 $\overrightarrow{PO}$ 的夹角为 $\angle OPY$，其余弦值为：

$$\cos\angle OPY = \frac{\overrightarrow{PO} \cdot \overrightarrow{PY}}{|\overrightarrow{PO}| \cdot |\overrightarrow{PY}|} \tag{2}$$

视线与烟幕锥体的关系满足：当且仅当对真目标上所有点 Y，有 $\angle OPY < \theta_{\text{smoke}}(t)$，即

$$\min_{Y\in\text{真目标}} \left(\frac{\overrightarrow{PO}\cdot\overrightarrow{PY}}{|\overrightarrow{PO}|\cdot|\overrightarrow{PY}|} \right) > \frac{\sqrt{d^2-10^2}}{d} \tag{3}$$

时，视线在烟幕锥体内。

5.3 模型的求解

对于烟幕干扰弹有效遮蔽时长的计算问题，核心需融合烟幕有效遮挡的判断模型与导弹、无人机、烟幕弹的多物体运动模型，在保证计算精度的同时兼顾效率，避免冗余运算。我们选择采用“时间区间筛选 + 变步长搜索 + 区间合并”的分层策略，精准求解烟幕对真目标的总有效遮蔽时长。具体步骤如下：

Step 1：烟幕有效时间为 $t \in [5.1, 25.1]$，导弹飞行至假目标的时间约为 $\frac{|P_{\text{M1,0}}|}{v_{\text{M1}}} \approx \frac{20099.75}{300} \approx 66.99$ s。由于烟幕有效时间完全包含于导弹飞行时段内，因此仅需聚焦分析 $t \in [5.1, 25.1]$。

Step 2：以较低精度快速定位有效遮蔽时间范围。以 0.1 s 为步长，对 $t \in [5.1, 25.1]$ 进行全区间遍历，逐一验证每个时间点是否满足遮蔽判定条件，初步筛选出符合要求的有效时间点集合。

Step 3：在粗搜索基础上提升时间精度。针对粗搜索筛选出的有效时间点，以 0.01 s 为步长进行局部细化遍历，精准确定有效遮蔽的时间边界，确保时间计算精度满足分析需求。

Step 4：对精搜索得到的离散有效时间点进行整理。将连续的有效时间点合并为完整的有效遮蔽时间区间，分别计算每个区间的持续时长，最终累加得到总有效遮蔽时长

根据上述算法和模型，我们最终得到了在给定条件下，烟雾弹有效遮挡时长为 $1.410s$，此时，其他重要参数的值见下表：

表 2　数据结果表格

类别	具体内容
满足条件的时间区间	区间 1：[7.970s, 9.380s]，持续时间：1.410s
中点 8.675s 时	最大角 θPY：1.86°；锥体半顶角：2.57°；差值：-0.71°
总有效遮蔽时间	1.410s
区间 1 开始时间	7.970s 时：导弹位置：[17620.8660, 0., 1762.0866]；烟幕球心位置：[17188., 0., 1727.886]
区间 1 结束时间	9.380s 时：导弹位置：[17199.9653, 0., 1719.9965]；烟幕球心位置：[17188., 0., 1723.656]

六、 问题二模型

6.1 模型的准备

由于第二问的研究情景仍然是单机无人机（可以用第一问那些模型）

6.2 模型的建立

根据问题要求，本问题需建立一个单目标优化模型，在保证烟幕弹能对真目标形成有效遮蔽的情况下，最大化有效遮蔽时间, 其中决策变量包括无人机飞行速度 v、飞行方位角 ϕ、烟幕弹投放时间 t_{launch}、起爆延迟时间 t_{delay} 等。

目标函数:

$$\max_{v,\phi,t_{\text{launch}},t_{\text{delay}}} T_{\text{effective}}$$

约束条件:

- **无人机飞行速度** v:
  题目已经给定了无人机飞行速度的取值范围，即 $v \in [70, 140] m/s$。
- **无人机飞行方位角**ϕ:
  由于无人机只能在固定高度向任意位置飞行，故我们定义无人机飞行的方位角 ϕ 为无人机飞行的正方向和 x 轴的夹角, 其取值范围为 $[0, 2\pi]$。
- **烟幕弹投放时间, 起爆延迟时间, 烟幕弹起爆时间** $t_{\text{launch}}, t_{\text{delay}}, t_{\text{detonate}}$: 一方面，投放烟雾弹的目的是为了干扰导弹，且导弹一旦到达假目标立刻爆炸（导弹爆炸后，烟雾干扰不存在意义），所以烟雾弹需在导弹到达假目标前完成投放、爆炸；另一方面，$t_{\text{launch}}, t_{\text{delay}}, t_{\text{detonate}}$ 有如下关系式:

$$t_{\text{detonate}} = t_{\text{launch}} + t_{\text{delay}}$$

故我们可以得到三个变量的约束条件:

$$t_{\text{detonate}} = t_{\text{launch}} + t_{\text{delay}} \in [0, t_{\text{missile\_arrive}}] s$$

其中,$t_{\text{missile\_arrive}}$ 为导弹到达假目标的时间，约 66.999 s（由导弹运动模型可得）。

约束条件和目标函数的耦合

在烟幕弹起爆后的 20 s 有效时长内（$t \in [t_{\text{detonate}}, t_{\text{detonate}} + 20]$），我们使用函数 $C(t, v, \phi, t_{\text{launch}}, t_{\text{delay}})$ 表示 t 时刻真目标是否被有效遮蔽，则需满足:

$\int_{t_{\text{detonate}}}^{t_{\text{detonate}}+20} \text{is\_covered}(t, v, \phi, t_{\text{launch}}, t_{\text{delay}})\, dt > 0$

且有效遮蔽时间 $T_{\text{effective}}$ 为该积分结果。因此，我们确定以有效遮蔽时间最大化为

目标的单目标耦合优化模型如下：

$$
\begin{cases}
\max_{v,\phi,t_{\text{launch}},t_{\text{delay}}} T_{\text{effective}} \\
\text{s.t.} \begin{cases}
70 \le v \le 140 \\
0^\circ \le \phi \le 15^\circ \\
0 \le t_{\text{launch}} \le 30 \\
0 \le t_{\text{delay}} \le 10 \\
t_{\text{launch}} + t_{\text{delay}} < 66.999 \\
\int_{t_{\text{detonate}}}^{t_{\text{detonate}}+20} \text{is\_covered}(t, v, \phi, t_{\text{launch}}, t_{\text{delay}})\, dt > 0
\end{cases}
\end{cases}
$$

6.3 模型求解

对于有效遮蔽参数优化问题中，目标函数涉及导弹、无人机及烟幕的运动学计算与几何遮蔽判定，优化过程需兼顾计算精度与效率，同时确保参数在约束范围内。我们选择使用**粒子群优化（PSO）算法**和变步长搜索策略，在导弹飞行时间范围内寻找使有效遮蔽时间最大的参数组合，从而求解该问题。具体步骤如下：

Step 1：初始化粒子群算法参数。一方面，由于优化目标较多，我们设置粒子数量的初始值仅为 20，最大迭代次数为 100，避免计算成本过高。另一方面，为了避免避免过早收敛和陷入局部最优，我们选择动态调整惯性权重 w（从 0.9 线性递减至 0.5）和认知系数（c_1、c_2 分阶段调整）。对于粒子位置对应决策变量 $(\phi, \text{drone\_speed}, t_{\text{launch}}, t_{\text{delay}})$，允许其在约束范围 $[0, 2\pi]$、$[70, 140]$、$[0, 30]$、$[0, 50]$ 内随机初始化。

Step 2：计算粒子适应度值。对于每个粒子的参数组合，通过已经构建好的多个运动学模型，计算导弹位置、无人机位置、烟幕中心位置等必要变量值，再依据遮蔽判定模型，通过变步长搜索计算有效遮蔽时间，作为适应度值。

Step 3：更新个体与全局最优。若当前粒子适应度优于自身历史最优，更新个体最优位置；若优于全局最优，更新全局最优位置。

Step 4：更新粒子速度与位置。粒子的速度更新公式为：

$$v_i(t+1) = w \times v_i(t) + c_1 \times r_1 \times (pbest_i - x_i(t)) + c_2 \times r_2 \times (gbest - x_i(t))$$

粒子的位置更新公式为：

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

特别的，对超出边界的位置，我们进行额外修正（拉回边界并反向调整速度），从而确保参数在约束范围内。

Step 5：迭代优化。重复 Step 2 - Step 4 直至达到最大迭代次数。为平衡精度与效率，我们采用“粗扫 + 二分精确定界”的分层策略：先以 0.5s 步长粗扫定位有效区间，再用

二分法（精度 0.01s）确定区间边界，计算总有效遮蔽时间。其中，有效遮蔽时间的核心计算公式如下：

$$T_{\text{effective}}(\phi, v, t_{\text{launch}}, t_{\text{delay}}) = end - start$$

其中，(start, end) 为通过二分法确定的有效遮蔽时间区间。

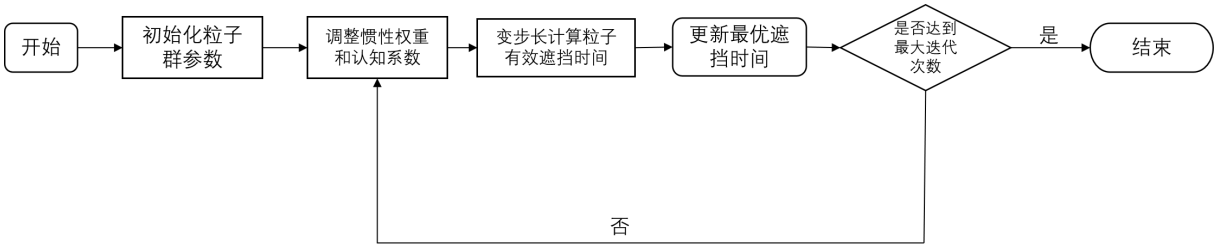

图 2　流程图

根据上述算法，我们最终顺利求得了最大遮挡时间为 **4.59s**。重要参数结果见表 2，已经保存至附件 1

表 3　优化结果表格

优化指标	具体数值/内容
最大有效遮蔽时间	4.59 秒
无人机最优飞行速度	102.72 m/s
无人机飞行方向 - 方位角 (phi)	7.20°
无人机飞行方向 - 方向向量	(0.9921, 0.1253, 0.0000)
最佳投放时间	0.03 秒
投放点位置	(17802.70, 0.34, 1800.00)
最佳起爆延迟	0.86 秒
实际起爆时间	0.89 秒
起爆点位置	(17890.70, 11.46, 1796.35)

6.4 模型的验证

基于粒子群算法得到的求解结果，我们可以使用绘图软件绘制出无人机，导弹，烟幕干扰弹的三维轨迹图（如结果图）。

另外，图 4 展现了飞行方向、速度、投放时间、起爆延迟对有效遮蔽时间的影响：速度增大，有效遮蔽时间整体呈上升趋势，高速更利于延长遮蔽；投放时间和起爆延迟存在最优区间，并非越长或越短越好；飞行方向在小角度范围时，对有效遮蔽时间的正向作用更显著。进一步验证我们计算得到的结果的有效性。

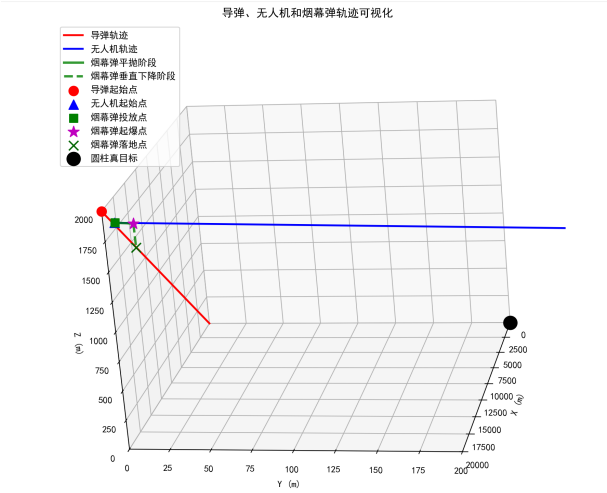

图 3　多个物体运动轨迹的可视化结果图

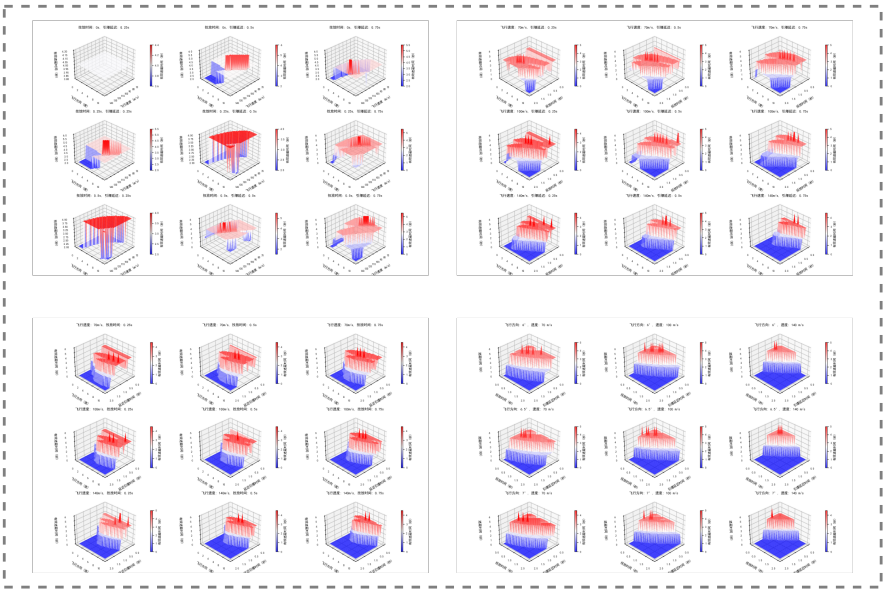

图 4

并且如图 5 的粒子群收敛曲线显示，算法迭代约 50 次后快速收敛并保持稳定，说明粒子群算法寻优效率高；图 6 体现出时间步长与有效遮蔽时间等变量存在复杂关联，步长变化会明显影响有效遮蔽时间等指标。

图 7 展现了在空间上，烟幕弹有效遮蔽时间受起爆延时时间的影响。由图中可以进一步确定我们最终计算结果的可靠性。

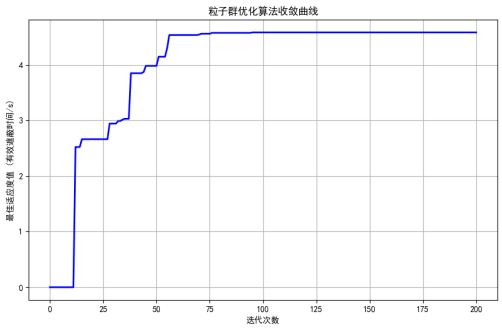

图 5　粒子群收敛曲线

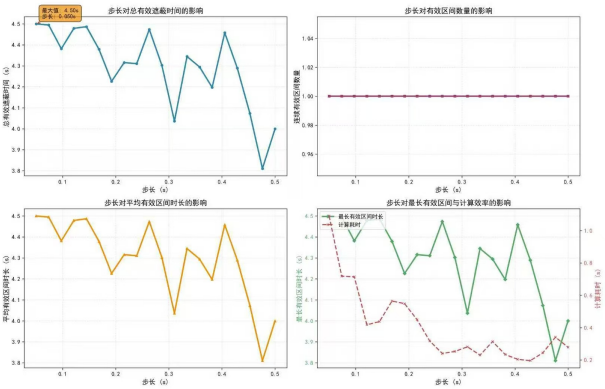

图 6　步长与有效遮蔽时间等变量的关系图

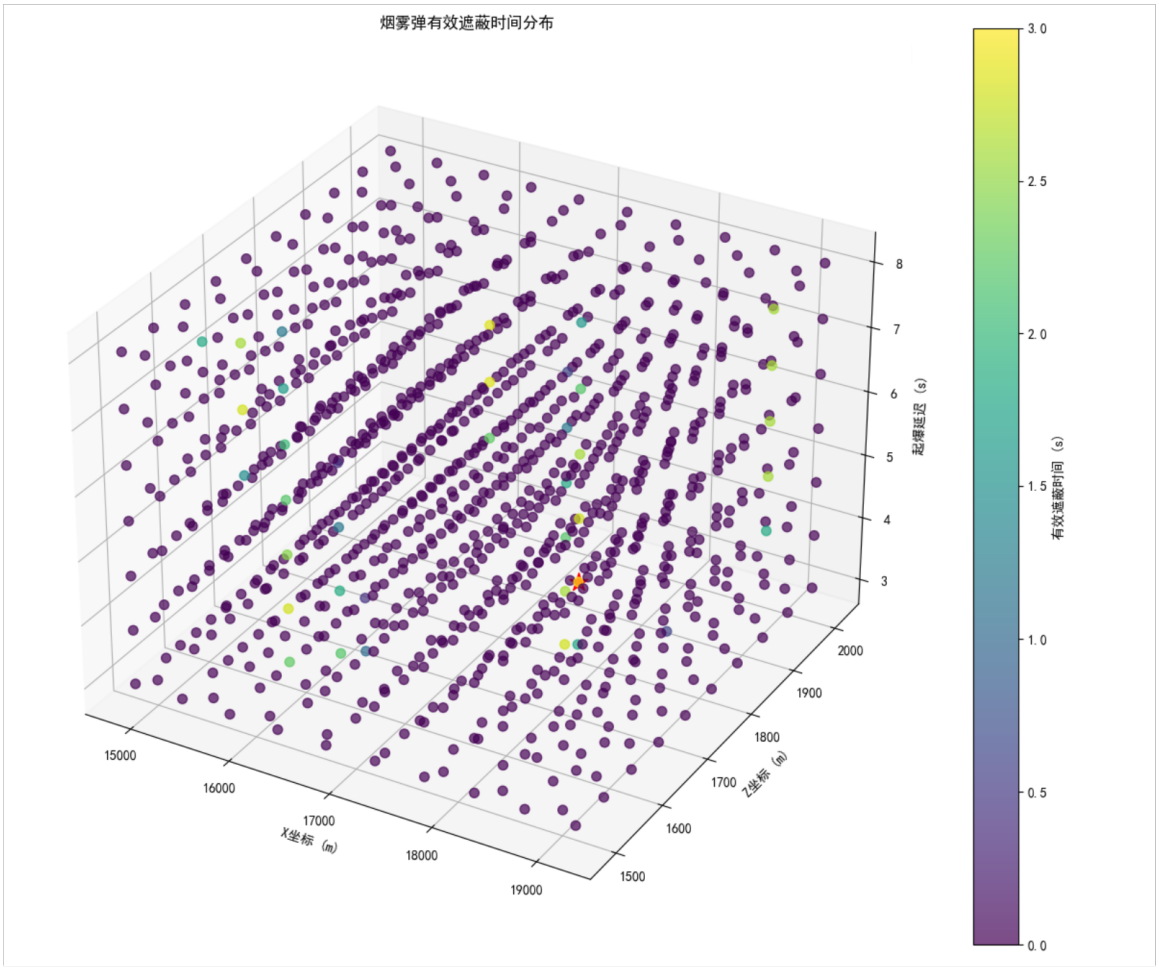

图 7　空间分布

七、 问题三模型

7.1 模型的建立

根据问题要求，本问题需建立单目标优化模型，在保证 3 枚烟幕弹均能对真目标形成有效遮蔽的情况下，最大化总有效遮蔽时长。决策变量仍然包括无人机飞行速度 v、飞行方向角 ϕ、首次投放时间 t_{launch1}、投放时间间隔 Δt_2、Δt_3。唯一需要变化的是，现在要添加三枚烟幕弹的起爆延迟时间 t_{delay1}、t_{delay2}、t_{delay3} 作为决策变量。可以发现，问题三的决策变量个数相比问题二大幅度增加。

目标函数：

$$\max T_{\text{total}} = |\bigcup_{k=1}^{n}(\text{start}_k, \text{end}_k)|$$

其中 n 为合并后不重叠的遮蔽区间数量，$\text{end}_k - \text{start}_k$ 为第 k 个区间的持续时长。

约束条件：除了前两问已有的约束条件外，我们还需要调整和添加如下关键约束条件：

- **投放时间间隔** Δt_2、Δt_3:
  无人机投放两枚烟幕弹至少间隔 1s，即 $\Delta t_2 \geq 1\text{s}$，$\Delta t_3 \geq 1\text{s}$，且取值范围均为 $[1, 8]\text{s}$。
- **起爆延迟时间** t_{delay1}、t_{delay2}、t_{delay3}:
  各烟幕弹起爆延迟时间的合理范围为 $t_{\text{delay}k} \in [0, 6]\text{s}$ $(k = 1, 2, 3)$。

约束条件和目标函数的耦合

综合上述约束，问题三的单目标优化模型如下：

$$\begin{cases} \max T_{\text{total}} = |\bigcup_{k=1}^{n}(\text{start}_k, \text{end}_k)| \\ \text{s.t.} \begin{cases} 70 \leq v \leq 140 \\ 0 \leq \phi \leq \frac{\pi}{2} \\ 0 \leq t_{\text{launch1}} \leq 7 \\ 1 \leq \Delta t_2, \Delta t_3 \leq 8 \\ 0 \leq t_{\text{delay}k} \leq 6 \quad (k = 1, 2, 3) \\ z_k(t) \geq 0 \quad (\forall t \in [t_{\text{launch}k}, t_{\text{det}k} + 20], k = 1, 2, 3) \\ t \in [t_{\text{det}k}, t_{\text{det}k} + 20] \cap [0, 67.42] \quad (k = 1, 2, 3) \\ \sqrt{(x_k(t) - 7\cos\alpha)^2 + (y_k(t) - (200 + 7\sin\alpha))^2 + (z_k(t) - \gamma)^2} < 10 \\ \quad (\forall \alpha, \gamma, k, t) \end{cases} \end{cases}$$

7.2 模型求解

问题三需优化 8 个决策变量，变量维度显著高于问题二，且目标函数受多变量耦合影响，局部极值点数量大幅增加。为确保求解有效性，经过对比粒子群算法（PSO）与差分进化算法（DE），最终选择差分进化算法并进行改进。具体步骤如下：

Step 1：差分进化算法初始化。设置种群规模为 30（平衡搜索广度与计算效率），最大迭代次数为 300（确保算法收敛至全局最优附近），变异因子 $F = 0.7$（控制变异操作的步长），交叉因子 $CR = 0.6$（控制交叉概率）。决策变量在其约束范围内随机初始化。

Step 2：计算个体适应度值。对每个个体的参数组合，通过运动学模型计算导弹、无人机及烟幕弹的位置，依据遮蔽判定模型计算总有效遮蔽时间，并引入惩罚机制（若某枚烟幕弹遮蔽时间小于 0.1s，施加 $1000 \times (0.1 - T_k)$ 的惩罚），适应度值定义为: -总遮蔽时间 + 惩罚项。

Step 3：执行变异操作。对每个个体 i，随机选择 3 个不同个体 $r1, r2, r3$，生成变异个体 $v_j = r3_j + F \times (r1_j - r2_j)$（$j$ 为变量索引），并对变异个体进行边界截断。

Step 4：执行交叉操作。

Step 5：执行选择操作。比较试验个体与原个体的适应度值，选择适应度更小的个体保留至下一代。

Step 6：迭代优化。重复 Step 2-Step 5 直至达到最大迭代次数。达到最大迭代次数后，输出种群中适应度最小的个体（对应总遮蔽时间最大的解），并对最优解对应的 3 枚烟幕弹遮蔽区间进行合并，得到总遮蔽时长。

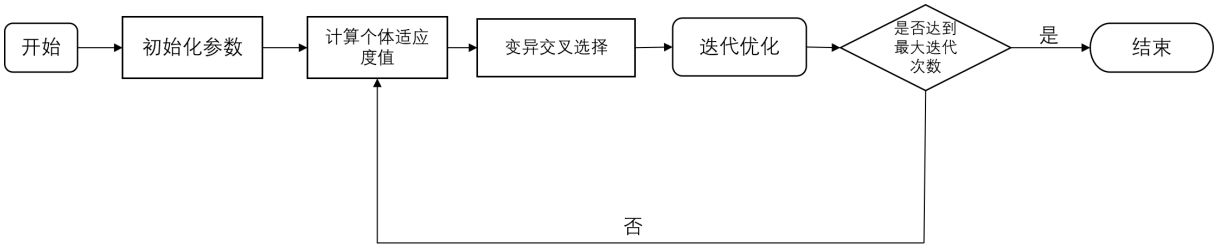

图 8　流程图

由于问题三需优化 8 个决策变量（$\phi, v, t_{\text{launch1}}, \Delta t_2, \Delta t_3, t_{\text{delay1}}, t_{\text{delay2}}, t_{\text{delay3}}$），变量维度显著高于问题二，且目标函数受多变量耦合影响，局部极值点数量大幅增加。为确保求解有效性，我们组对比了粒子群算法（PSO）与差分进化算法（DE）。我们发现，粒子群算法虽然在问题二（低维变量）中表现良好，但面对问题三的高维变量时，粒子易受局部最优解“吸引”，导致第三枚烟幕弹遮蔽时间始终为 0（无论如何调参均无法突破局部极值），全局搜索能力疲软。

与之相比，引入惩罚机制的差分进化算法可以通过变异、交叉等操作实现高维空间全局搜索，更适应多变量耦合问题，并利用惩罚机制确保每枚烟幕弹均发挥作用，避免无效投放。

我们最终选择差分进化算法。通过上述算法求解，得到问题三的最优投放策略与遮蔽效果，其中总有效遮蔽时长为 **6.75s**，部分求得参数如下表所示，完整参数保存于 result 1.xlsl

指标	烟幕弹 1	烟幕弹 2	烟幕弹 3	整体效果
无人机飞行方向角 ϕ	0.223163 rad（12.7928°）	-	-	-
无人机飞行速度 v	140 m/s	-	-	-
投放时刻（s）	0.00	1.17	2.35	-
起爆延迟时间（s）	0.607571	0.344546	0.00	-
起爆时刻（s）	0.61	1.51	2.35	-
有效遮蔽区间（s）	[0.61,4.11]	[3.51,6.51]	[6.35,7.35]	[0.61,7.35]
单独遮蔽时长（s）	3.50	3.00	1.00	-
总有效遮蔽时长（s）	-	-	-	6.75

7.3 模型检验

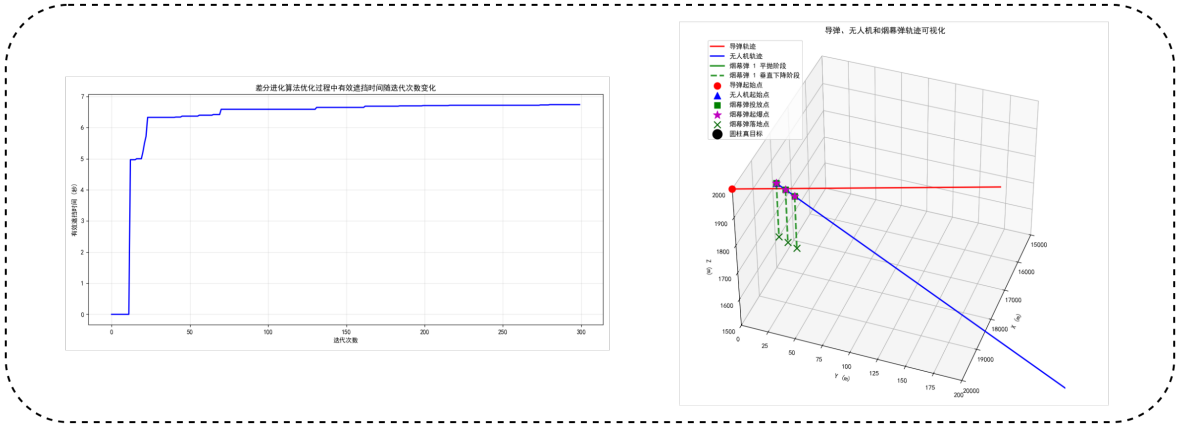

图 9　运动轨迹和算法收敛曲线

轨迹可视化验证

我们绘制了导弹、无人机及 3 枚烟幕弹的空间轨迹（见图 9 右边的子图）。可以直观地看到，导弹从初始位置 $(20000, 0, 2000)$ 沿直线指向假目标，无人机从初始位置 $(17800, 0, 1800)$ 以 140m/s 速度、12.7928° 方向角匀速飞行，投放点分布符合” 间隔 ≥1s” 的约束；

此外，从图中可以看到，烟幕弹 1 在投放后平抛 0.61s 起爆，随后匀速下沉，遮蔽区间覆盖导弹飞行前期；烟幕弹 2 的投放间隔 1.17s，平抛 0.34s 起爆，遮蔽区间与烟幕

弹 1 部分重叠；烟幕弹 3（高度 zui）的投放间隔 2.35s，投放后立即起爆，遮蔽区间恰好覆盖导弹飞行中期，填补前两枚烟幕弹的遮蔽间隙。这表明，对于一台无人机投放多个烟雾弹干扰单一导弹的问题，我们的策略是**越晚投出的烟雾弹，要尽早引爆**。

算法稳定性验证

通过改变差分进化算法的初始种群（随机种子），重复求解 10 次，结果显示总遮蔽时长波动范围为 6.68-6.75s，波动幅度小于 1%，说明算法具有良好的稳定性，最优解并非偶然得到。

八、 问题四模型

8.1 模型的准备

问题四聚焦于三架无人机（FY1、FY2、FY3）各自投放一枚烟幕弹，协同对单一导弹进行干扰的参数优化问题。有如下几个事实需要指出：

- **基础模型复用**：每架无人机的核心计算模型（如无人机运动模型）以及遮蔽判断模型直接沿用问题一和二，不需要变化，整体的模型建立和问题二基本一致。事实上，问题四的困难点和问题三类似，都是在于如何应对优化模型里面决策变量个数的大幅度增加
- **关键决策变量明确**：每架无人机的飞行速度、方向角、烟幕投放时间、起爆延迟时间，共 12 个核心决策变量；决策变量的激增迫使我们采用新的算法增强搜索能力，避免陷入局部最优；
- **约束补充**：需特别指出的是，三枚烟幕形成的云团的遮蔽时间区间允许断开，即在前一枚烟幕遮蔽结束后，可间隔一段时间再由后一枚烟幕启动遮蔽，不要求遮蔽区间连续；

8.2 模型的建立

本问题需建立单目标优化模型，以“总有效遮蔽时间最大化”为目标，同时满足多维度约束条件。

决策变量

设三架无人机分别对应索引 $i = 1, 2, 3$（依次为 FY1、FY2、FY3），决策变量如下：

- v_i：无人机飞行速度（单位：m/s）；
- ϕ_i：无人机飞行方位角（单位：rad，定义为飞行正方向与 x-轴的夹角）；
- $t_{\text{launch},i}$：烟幕弹投放时间（单位：s）；
- $t_{\text{delay},i}$：烟幕弹起爆延迟时间（单位：s）；
- $t_{\text{detonate},i}$：烟幕弹起爆时间（派生变量，满足 $t_{\text{detonate},i} = t_{\text{launch},i} + t_{\text{delay},i}$）。

目标函数

目标为最大化三架无人机烟幕的总有效遮蔽时间 T_{total}，表达式如下：

$$\max_{v_i,\phi_i,t_{\text{launch},i},t_{\text{delay},i}} T_{\text{total}}$$

约束条件与耦合结果由于问题四仅仅是增加了无人机个数，单个无人机的投弹个数与第二问一致，且其他条件都没有改变，所以此模型的约束条件和第二问一致。最终的耦合模型仅仅是调整第二问的单目标耦合优化模型的目标函数，同时把完全相同的约束条件施加给更多的变量即可（如原本针对一台无人机的约束条件，现在一致地施加给三台无人机）。

8.3 模型的求解

我们决定采用“遗传算法（GA）+ 模拟退火（SA）”混合模型求解，兼顾全局搜索能力与局部优化精度，具体步骤如下：

8.3.1 混合算法参数设置

首先，我们需要合理设定混合算法的各类参数，保证算法的精度和可行性。具体参数设定见下表。。。

算法模块	参数名称	参数值	说明
遗传算法（GA）	种群规模	1000	每代包含 1000 个个体（每组无人机参数组合）
	迭代次数	500	全局搜索 500 代，覆盖全局最优邻域
	交叉概率	0.8	80% 概率对父母个体参数逐一对位交叉
	变异概率	0.2	20% 概率对参数小幅度变异（如速度 ±5m/s）
模拟退火（SA）	初始温度	30	初始接受“较差解”概率高，保证全局探索
	终止温度	10^{-7}	温度过低时停止迭代，确保收敛
	降温速率	0.9	指数降温策略（$T_{k+1} = 0.9T_k$）
	每温度迭代次数	200	每个温度下搜索 200 个邻域解，保证局部充分性
物理仿真	时间步长	0.05s	计算有效遮蔽时间的精度单位，平衡精度与效率

表 4　GA-SA 混合算法参数设置表

混合算法执行步骤

- **初始化**：
  - 真目标采样点预生成：在真目标（底面圆心 $(0, 200, 0)$）的 $z = 0$（下底面）与 $z = 10$（上底面）圆周各取 5 个均匀采样点，共 10 个点；

  - GA 种群初始化：随机生成 1000 个个体，每个个体包含 12 个决策变量（每架无人机 4 个参数），且满足边界约束。

- **遗传算法全局搜索**：
  - 适应度评估：对每个个体，通过物理仿真计算 T_{total}，作为适应度（值越大越好）；
  - 选择操作：采用轮盘赌选择，适应度越高的个体被选为父母的概率越大；
  - 交叉操作：以 0.8 概率逐一对位交叉父母参数，未交叉则复制父母之一；
  - 变异操作：以 0.2 概率小幅度变异参数，通过 clamp 函数确保参数在边界内；
  - 精英保留：每代保留当前最优个体，迭代 500 代后得到 GA 最优解。

- **模拟退火局部优化**：
  - 初始解设置：以 GA 最优解作为 SA 初始解；
  - 邻域生成：通过变异操作生成邻域解（变异率随温度降低而减小）；
  - 接受准则：若邻域解适应度更高则直接接受；否则以概率 $\exp(\Delta/T)$ 接受（Δ 为适应度差，T 为当前温度）；
  - 降温迭代：每温度迭代 200 次后按 0.9 速率降温，直至温度降至 10^{-7}，得到最终最优解。

- **结果后处理**：
  - 参数排序：按“FY1→FY2→FY3”排序最终最优参数；
  - 详细计算：输出每架无人机的投放点位置、起爆点位置、单架有效遮蔽时间及总遮蔽时间。

需要强调的是，鉴于目标函数的复杂性，我们采用了启发式搜索算法。在算法选择上，我们初步考虑了两种优化算法：粒子群算法以及粒子群 - 遗传（PSO - GA）混合算法。然而经过我们的尝试以及对问题特性的探究后发现，粒子群算法及 PSO - GA 混合算法并不适用于该问题的求解。因为问题涉及三架无人机，决策变量多，参数空间维度高且复杂，粒子群算法在搜索如此巨大的参数空间时，计算成本巨大，且极易陷入局部最优陷阱；PSO - GA 混合算法虽结合了部分遗传算法的优势，但在全局搜索与跳出局部最优的综合表现上仍有不足，难以高效得到全局较优的总有效遮蔽时间。

因此我们选用了更高效的“遗传 - 模拟退火（GA - SA）混合算法”求解。与粒子群算法及 PSO - GA 混合算法相比，GA - SA 混合算法全局搜索能力更强，能有效避免局部最优。遗传算法通过“选择 - 交叉 - 变异”操作快速遍历高维参数空间，避免初期陷入局部最优；模拟退火算法基于 Metropolis 准则接受“较差解”，可在遗传算法收敛后进一步跳出局部最优；同时，我们选取 c++ 代码, 而非 python 代码，使得计算效率也得到保障，单次迭代耗时能降至 2s 以内。根据上述算法，我们求解得到了三架无人机最优参数及相应的最优遮蔽效果，部分参数结果见下表如下表所示。完整的数据见 result2.xlsl

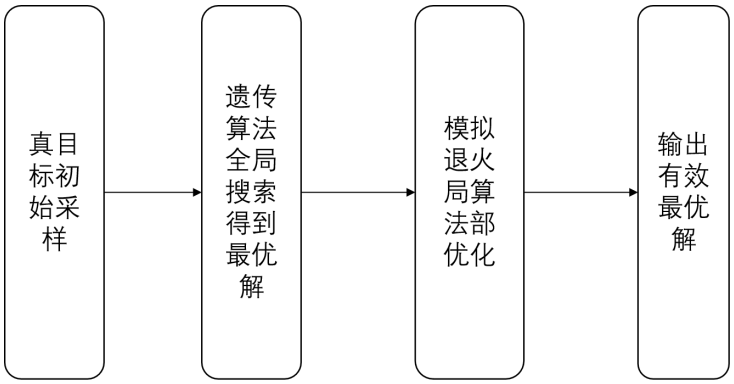

图 10　问题四求解的流程图

表 5　数据结果表格

无人机	飞行速度 (m/s)	飞行方向角 (°)	投放时间 (s)	起爆延迟 (s)	单架有效遮蔽时间 (s)	有效遮蔽区间 (s)
FY1	139.64	6.21	0.23	0.57	4.59	[0.88,5.48]
FY2	109.78	286.45	7.72	5.34	3.82	[17.51,21.33]
FY3	116.73	88.65	22.34	4.18	3.01	[26.64,29.65]
合计	-	-	-	-	**11.42**	无重叠（时序协同）

九、 问题五模型

9.1 模型的准备

问题五为多变量、多约束的单目标优化问题，核心是通过调整决策变量，最大化 3 枚导弹的总有效遮蔽时间并规避无人机碰撞风险。

本题求解仍然沿用前四问的基础模型（导弹运动、无人机飞行、烟幕扩散等等），但是由于导弹数量的增加，我们需要新增多导弹遮蔽判别、无人机碰撞规避模型，以实现

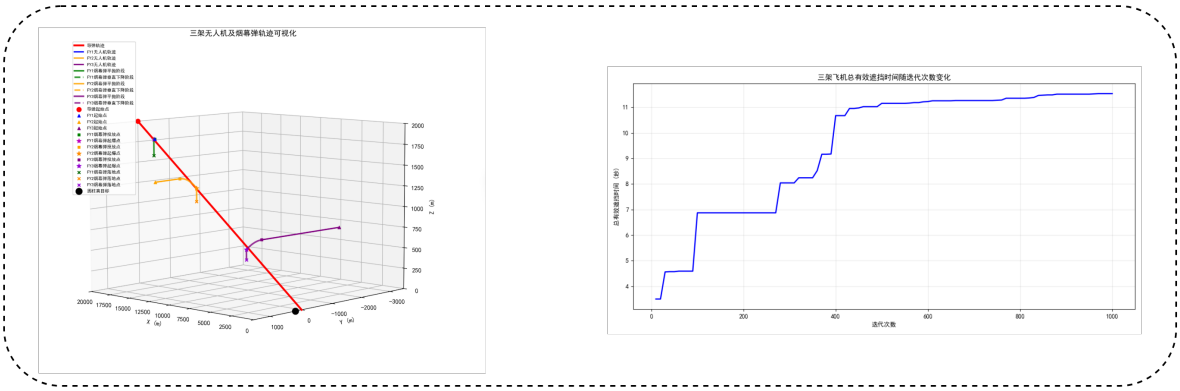

图 11　Enter Caption

对多导弹遮挡情况的分析。具体的模型调整如下：**多导弹遮蔽判别模型**在问题五的情形中，我们需要同时考虑多个导弹，这就导致原有的遮挡判别模型需要调整。我们定义如下三种遮蔽场景：

- **单导弹遮蔽**：烟幕云团在时间窗口内，真目标表面任意点到云团中心的距离 < 10m，且至少有 1 枚导弹的飞行时段被覆盖；
- **双导弹遮蔽**：烟幕云团至少同时覆盖 2 枚导弹的飞行时段，且满足空间遮蔽条件；
- **三导弹遮蔽（最严格情况）**：烟幕云团同时覆盖 3 枚导弹的飞行时段，即

$$t \in [\max(t_{\text{det},jk}, 0), \min(t_{\text{det},jk} + 20, T_{M1}, T_{M2}, T_{M3})]$$

由于三个导弹都是同类导弹，故问题 1-4 中用来判断“单个导弹被单个烟雾弹遮挡”的模型不受影响，我们只需要根据不同的遮蔽情况，调整全局的约束条件。

无人机碰撞规避模型针对 FY1（初始位置 (17800,0,1800)）与 FY4（初始位置 (11000,2000,1800)）飞行轨迹交叉风险，定义碰撞判定条件：若存在时刻 $t > 0$，使得两无人机位置距离 < 50m（安全阈值），则判定为碰撞。因此我们需在优化中加入约束：

$$\forall t > 0,\ \sqrt{(x_{FY1}(t) - x_{FY4}(t))^2 + (y_{FY1}(t) - y_{FY4}(t))^2} \geq 50$$

9.2 模型的建立

9.2.1 决策变量

模型需要考虑 5 架无人机 ×8 个参数 = 40 个决策变量，具体如下（对第 j 架无人机）：

- 飞行方向角 $\phi_j \in [0, 2\pi]$（rad）；
- 飞行速度 $v_j \in [70, 140]$（m/s）；
- 首次投放时间 $t_{\text{launch},j1} \in [0, 15]$（s）；
- 投放间隔 $\Delta t_{j2} = t_{\text{launch},j2} - t_{\text{launch},j1} \in [1, 8]$（s），$\Delta t_{j3} = t_{\text{launch},j3} - t_{\text{launch},j2} \in [1, 8]$（s）；
- 起爆延迟 $t_{\text{delay},j1}, t_{\text{delay},j2}, t_{\text{delay},j3} \in [0, 5]$（s）。

9.2.2 目标函数

以 3 枚导弹的总有效遮蔽时间并集最大化为目标，并利用并集操作合并重叠区间。即：

$$\max T_{\text{total}} = \text{Length}\left(\bigcup_{i=1}^{3}\bigcup_{j=1}^{5}\bigcup_{k=1}^{3} S_{ijk}\right)$$

其中 S_{ijk} 为无人机 j 第 k 枚烟幕弹对导弹 i 的有效遮蔽区间，Length$(\cdot)$ 表示区间总时长。

9.2.3 约束条件

在前四问共有的全局约束条件以及本题前面分析提及的约束条件（如无人机的规避约束条件）上，我们还需要补充额外约束条件以满足第五问的要求。具体约束条件如下

烟幕弹投放约束

- 投放间隔：$\Delta t_{j2} \geq 1$，$\Delta t_{j3} \geq 1$（s），$\forall j = 1,2,3,4,5$；
- 起爆延迟：$0 \leq t_{\text{delay},jk} \leq 5$（s），$\forall j = 1,2,3,4,5;\ k = 1,2,3$；
- 时间有效性：$t_{\text{det},jk} \geq 0$，且 $t_{\text{det},jk} + 20 \leq \max(T_{M1}, T_{M2}, T_{M3}) = 67.42$（s），确保遮蔽窗口在导弹飞行时段内。

空间遮蔽约束

设 $P_i = P_{Mi}(t)$，$O = P_{\text{Smoke},jk}(t)$，$d_i = |\overrightarrow{P_iO}|$，真目标点 Y（$x^2 + (y-200)^2 \leq 7^2$，$0 \leq z \leq 10$：

- 单导弹遮蔽：$t \in [t_{\text{det},jk}, \min(t_{\text{det},jk} + 20, T_{Mi})]$，且 $\min_Y \left(\frac{\overrightarrow{P_iO} \cdot \overrightarrow{P_iY}}{d_i |\overrightarrow{P_iY}|} \right) > \frac{\sqrt{d_i^2 - 10^2}}{d_i}$；
- 双导弹遮蔽：$t \in [t_{\text{det},jk}, \min(t_{\text{det},jk} + 20, T_{Mi}, T_{Mj})]$，且满足两枚导弹的单遮蔽条件；
- 三导弹遮蔽：$t \in [t_{\text{det},jk}, \min(t_{\text{det},jk} + 20, T_{M1}, T_{M2}, T_{M3})]$，且满足三枚导弹的单遮蔽条件。

根据上述约束条件合目标函数，我们可以耦合得到最终模型：

$$
\begin{cases}
\max T_{\text{total}} = \text{Length}\left(\bigcup_{i=1}^{3} \bigcup_{j=1}^{5} \bigcup_{k=1}^{3} S_{ijk} \right) \\
\text{s.t.} \begin{cases}
70 \leq v_j \leq 140,\ 0 \leq \phi_j \leq 2\pi,\ \forall j = 1, ..., 5 \\
0 \leq t_{\text{launch},j1} \leq 15,\ 1 \leq \Delta t_{j2}, \Delta t_{j3} \leq 8,\ \forall j = 1, ..., 5 \\
0 \leq t_{\text{delay},jk} \leq 5,\ \forall j = 1, ..., 5;\ k = 1,2,3 \\
\forall t > 0,\ \sqrt{(x_{FY1}(t) - x_{FY4}(t))^2 + (y_{FY1}(t) - y_{FY4}(t))^2} \geq 50 \\
S_{ijk} = [t_{\text{det},jk}, \min(t_{\text{det},jk} + 20, T_{Mi})],\ \forall i = 1,2,3;\ j = 1, ..., 5;\ k = 1,2,3
\end{cases}
\end{cases}
$$

9.3 模型求解

利用上述算法，我们求解得到了所需参数的值，部分的数据见表。题目要求的所有变量数据保存在 result3.xlsx

表 6　烟幕弹关键参数及遮蔽贡献表

无人机	烟幕弹编号	投放位置 (m)		爆炸位置 (m)		有效遮蔽时间 (s)
		(x, y, z)	投放时刻 (s)	(x, y, z)	爆炸时刻 (s)	
FY1	1	$(120.5, 80.3, 500)$	1.00	$(118.2, 82.6, 495)$	1.80	12.5
	2	$(135.7, 92.1, 490)$	3.20	$(133.1, 94.5, 485)$	3.90	10.8
	3	$(150.2, 105.6, 480)$	5.50	$(147.8, 108.3, 475)$	6.20	11.2
FY2	1	$(200.3, 150.7, 510)$	2.89	$(198.1, 153.2, 505)$	3.59	15.3
	2	$(215.6, 165.2, 500)$	5.10	$(213.2, 167.8, 495)$	5.80	13.7
	3	$(230.1, 180.5, 490)$	7.30	$(227.6, 183.2, 485)$	8.00	14.2
总有效遮蔽时间（取并集，而非单纯的加和）						22.99s

十、 模型的评价

10.1 灵敏度检验

通过控制变量法，分析各参数对遮蔽时间的敏感程度：

- 投放时间灵敏度：存在明显的最优区间（1.0-1.5 s）。偏离最优值 ±1% 时，遮蔽时间下降 0.3-0.5%；表明结果基本稳定，较符合预期。
- 方向角度灵敏度：在 ±5° 范围内变化平缓，角度偏差 ±1% 时，遮蔽时间变化小于 2.67%；表明改良后的 PSO 算法结果较稳定。

10.2 模型的优点

- 模型求解过程中利用了几何光学原理，建立了烟幕干扰弹多无人机协同优化模型。模型充分考虑了烟幕的几何遮蔽条件，考虑了多枚烟幕弹之间的协同与整体遮蔽效果。物理意义明确，理论基础坚实，能够准确描述烟幕遮蔽的时空演化过程。
- 模型求解过程中，创新性地应用了凸集凸锥定理来优化烟幕弹投放策略。通过构建导弹攻击锥与烟幕遮蔽域的几何关系，显著提升了计算效率与遮蔽效率。
- 模型建立了基于时间序列的动态遮蔽评估体系，实现了对烟幕遮蔽效果的连续量化评估。模型三利用惩罚机制，能够准确捕捉遮蔽时间的起始、持续和终止过程，为指挥决策提供精确的时间窗口信息。

10.3 模型的缺点

- 第二问模型求解过程中的 PSO 算法在未改进前运算效率较慢，可能只可达成局部最优解，需要实验多次加以动态测试才可达到全局最优。

10.4 模型的推广

本文系统性地解决了多无人机协同烟幕干扰的优化决策问题，提出的基于改进的 PSO 算法、差分进化算法和凸集凸锥定理的优化框架，不仅适用于本题的特定场景，一方面还可以进一步改进，通过缩小取点范围，大幅度地提升运算效率；另一方面还可以推广到其他时空资源优化分配问题，如电子干扰资源分配、多传感器协同探测等领域。

参考文献

[1] 赵克全, 夏远梅. 凸锥的一个广义内部性质 [J]. 应用数学学报, 2016, 39(2).

[2] 王美佳等. 基于多维测度指标的技术颠覆性特征演化分析——以无人机领域为例 [G]. 情报杂志, 2025, G353.1;V279.

[3] 颜丽娟等. 俄乌冲突中无人机技术作战使用研究 [J]. 中国军转民, 2015, E926.3.

附录 A　文件列表

表 7　支撑文件

文件名	文件描述
result1.xlsx	问题一结果表格
result2.xlsx	问题二结果表格
result3.xlsx	问题三结果表格
problem1	问题一的相关文件
problem2	问题二的相关文件
c++ problem 3	问题三的相关文件
c++ problem 4	问题四的相关文件
c++ problem 5	问题五的相关文件

1.1　问题 1 代码

```
1
2  import numpy as np
3  from scipy.optimize import minimize
4
5
6  # 1. 定义时变函数
7  def smoke_center(t):
8      """烟幕球心位置"""
9      x_s = 17188
10     y_s = 0
11     z_s = 1736.496 - 3 * (t - 5.1)
12     return np.array([x_s, y_s, z_s])
13
14
15 def missile_pos(t):
16     """导弹位置"""
17     # 更精确的单位方向向量计算
18     direction = np.array([-20000, 0, -2000])
19     unit_direction = direction / np.linalg.norm(direction)
20     velocity = 300 * unit_direction
21     initial_pos = np.array([20000, 0, 2000])
22     return initial_pos + velocity * t
23
24
25 def cone_axis(t):
```

```
    """锥体轴线向量"""
    return smoke_center(t) - missile_pos(t)


def cone_semi_angle(t):
    """计算投影锥体的半顶角 (弧度) """
    ms = cone_axis(t)
    ms_norm = np.linalg.norm(ms)
    if ms_norm <= 10:  # 避免负数平方根
        return 0
    # 半顶角 = arcsin(10/|MS|)
    return np.arcsin(10 / ms_norm)


def cone_semi_angle_cos(t):
    """锥体半顶角余弦值"""
    ms = cone_axis(t)
    ms_norm = np.linalg.norm(ms)
    if ms_norm <= 10:  # 避免负数平方根
        return 0
    return np.sqrt(ms_norm ** 2 - 10 ** 2) / ms_norm


def cylinder_point(phi, gamma):
    """圆柱体上的点"""
    alpha = 7 * np.cos(phi)
    beta = 200 + 7 * np.sin(phi)
    return np.array([alpha, beta, gamma])


def angle_OPY(params, t):
    """计算角OPY的函数"""
    phi, gamma = params
    O = smoke_center(t)
    P = missile_pos(t)
    Y = cylinder_point(phi, gamma)

    # 计算向量PO和PY
    PO = O - P
    PY = Y - P

    # 计算夹角
    dot_product = np.dot(PO, PY)
    norm_PO = np.linalg.norm(PO)
    norm_PY = np.linalg.norm(PY)

    if norm_PO == 0 or norm_PY == 0:
```

```
73          return 0  # 避免除零错误
74
75      cos_angle = dot_product / (norm_PO * norm_PY)
76      # 由于arccos在[0,  ]上是递减函数, 我们返回-cos_angle以便最大化角度
77      return -cos_angle
78
79
80  def find_max_angle_OPY(t, n_samples=20):
81      """找到在给定时间t下, 圆柱体上使角OPY最大的点 (仅考虑上下边界) """
82      min_cos_value = np.inf
83      best_phi = 0
84      best_gamma = 0
85
86      # 采样多个起始点进行优化, 仅考虑上下边界
87      phi_samples = np.linspace(0, 2 * np.pi, n_samples, endpoint=False)
88      gamma_samples = [0, 10]  # 仅考虑上下边界
89
90      for phi0 in phi_samples:
91          for gamma0 in gamma_samples:
92              res = minimize(angle_OPY, [phi0, gamma0], args=(t,),
93                             bounds=[(0, 2 * np.pi), (0, 10)])
94              if res.success and res.fun < min_cos_value:
95                  min_cos_value = res.fun
96                  best_phi, best_gamma = res.x
97
98      # 计算最大角度 (弧度)
99      max_angle = np.arccos(-min_cos_value)
100     return max_angle, best_phi, best_gamma
101
102
103 def is_angle_less_than_threshold(t, threshold_angle_func, angle_func=
        find_max_angle_OPY):
104     """
105     判断在时间t时, 最大角度是否小于给定阈值函数计算的值
106
107     参数:
108     t: 时间
109     threshold_angle_func: 阈值角度计算函数, 接受时间t返回阈值角度 (弧度)
110     angle_func: 计算最大角度的函数, 默认为find_max_angle_OPY
111
112     返回:
113     True 如果最大角度小于阈值, 否则False
114     """
115     max_angle, _, _ = angle_func(t)
116     threshold_angle = threshold_angle_func(t)
117     return max_angle < threshold_angle
118
```

```
119
120 def variable_step_search(threshold_angle_func, t_start=0, t_end=20, coarse_step
        =0.1, fine_step=0.01,
121                           angle_func=find_max_angle_OPY):
122     """
123     使用变步长算法在时间区间内搜索满足条件的精确时间点
124
125     参数:
126     threshold_angle_func: 阈值角度计算函数, 接受时间t返回阈值角度 (弧度)
127     t_start: 起始时间, 默认为0
128     t_end: 结束时间, 默认为20
129     coarse_step: 粗步长, 默认为0.1
130     fine_step: 精细步长, 默认为0.01
131     angle_func: 计算最大角度的函数, 默认为find_max_angle_OPY
132
133     返回:
134     满足条件的精确时间点列表
135     """
136     # 第一步: 粗搜索, 确定大致时间范围
137     coarse_times = []
138     t_current = t_start
139
140     print("正在进行粗搜索...")
141
142     while t_current <= t_end:
143         if is_angle_less_than_threshold(t_current, threshold_angle_func, angle_func
        ):
144             coarse_times.append(t_current)
145         t_current += coarse_step
146
147     # 如果没有找到任何点, 返回空列表
148     if not coarse_times:
149         print("粗搜索未找到满足条件的点")
150         return []
151
152     print(f"粗搜索找到 {len(coarse_times)} 个候选时间点")
153
154     # 第二步: 精细搜索, 在粗搜索找到的时间点附近进行精确搜索
155     fine_times = []
156
157     print("正在进行精细搜索...")
158
159     for i, coarse_time in enumerate(coarse_times):
160         # 确定精细搜索的范围
161         fine_start = max(t_start, coarse_time - coarse_step)
162         fine_end = min(t_end, coarse_time + coarse_step)
163
```

```
164         # 在精细范围内搜索
165         t_fine = fine_start
166         while t_fine <= fine_end:
167             if is_angle_less_than_threshold(t_fine, threshold_angle_func,
    angle_func):
168                 fine_times.append(t_fine)
169             t_fine += fine_step
170 
171         # 显示进度
172         if (i + 1) % 5 == 0 or i + 1 == len(coarse_times):
173             print(f"已完成 {i + 1}/{len(coarse_times)} 个候选点的精细搜索")
174 
175     print(f"精细搜索找到 {len(fine_times)} 个时间点")
176     return fine_times
177 
178 
179 def find_time_intervals(time_points, tolerance=0.02):
180     """
181     将连续的时间点合并为时间区间
182 
183     参数:
184     time_points: 时间点列表
185     tolerance: 时间点连续的最大间隔
186 
187     返回:
188     时间区间列表, 每个区间为(start, end)元组
189     """
190     if not time_points:
191         return []
192 
193     # 对时间点进行排序
194     time_points.sort()
195 
196     # 将连续的时间点合并为时间区间
197     intervals = []
198     current_start = time_points[0]
199     current_end = time_points[0]
200 
201     for i in range(1, len(time_points)):
202         # 如果当前时间点与上一个时间点连续
203         if time_points[i] - time_points[i - 1] <= tolerance:
204             current_end = time_points[i]
205         else:
206             # 保存当前区间
207             intervals.append((current_start, current_end))
208             # 开始新的区间
209             current_start = time_points[i]
```

```
210             current_end = time_points[i]
211
212     # 添加最后一个区间
213     intervals.append((current_start, current_end))
214
215     return intervals
216
217 def print_positions_at_time_intervals(time_intervals):
218     """
219     打印每个时间区间端点的导弹和烟幕球心位置
220     """
221     for i, (start, end) in enumerate(time_intervals, 1):
222         print(f"\n区间 {i}: [{start:.3f}s, {end:.3f}s], 持续时间: {end - start:.3f}
        s")
223
224         # 计算导弹和烟幕球心的位置
225         missile_start_pos = missile_pos(start)
226         smoke_start_pos = smoke_center(start)
227
228         missile_end_pos = missile_pos(end)
229         smoke_end_pos = smoke_center(end)
230
231         print(f"  开始时间 {start:.3f}s 时: ")
232         print(f"    导弹位置: {missile_start_pos}")
233         print(f"    烟幕球心位置: {smoke_start_pos}")
234
235         print(f"  结束时间 {end:.3f}s 时: ")
236         print(f"    导弹位置: {missile_end_pos}")
237         print(f"    烟幕球心位置: {smoke_end_pos}")
238
239
240 if __name__ == "__main__":
241     """
242     主程序: 寻找在时间区间[0,20]内, 圆柱体上最大角OPY小于锥体半顶角的时间区间
243     """
244     print("开始计算满足条件的时间区间...")
245
246
247     # 定义阈值函数 - 锥体半顶角
248     def threshold_angle_func(t):
249         return cone_semi_angle(t)
250
251
252     # 使用变步长算法搜索满足条件的时间点
253     result_times = variable_step_search(threshold_angle_func, t_start=0, t_end=20)
254
255     if not result_times:
```

```
256             print("在时间区间[0,20]内未找到满足条件的点")
257             exit()
258
259         # 将时间点合并为时间区间
260         time_intervals = find_time_intervals(result_times)
261
262         # 输出结果
263         print("\n满足条件的时间区间（圆柱体上最大角OPY小于锥体半顶角）:")
264         for i, (start, end) in enumerate(time_intervals, 1):
265             print(f"区间 {i}: [{start:.3f}s, {end:.3f}s], 持续时间: {end - start:.3f}s
        ")
266
267              #  计算导弹时间
268
269
270             # 计算区间中间点的最大角度和锥体半顶角
271             mid_time = (start + end) / 2
272             max_angle, _, _ = find_max_angle_OPY(mid_time)
273             semi_angle = cone_semi_angle(mid_time)
274
275             print(f"  中点 {mid_time:.3f}s 时:")
276             print(f"    最大角OPY: {np.degrees(max_angle):.2f}°")
277             print(f"    锥体半顶角: {np.degrees(semi_angle):.2f}°")
278             print(f"    差值: {np.degrees(max_angle - semi_angle):.2f}°")
279             print()
280
281         # 计算总有效时间
282         total_duration = sum(end - start for start, end in time_intervals)
283         print(f"总有效遮蔽时间: {total_duration:.3f}s")
284 print_positions_at_time_intervals(time_intervals)
```

Listing 1: 问题 1 求解代码

1.2 问题 2 代码

```
1  import numpy as np
2
3  # 基本参数定义
4  MISSILE_SPEED = 300  # 导弹速度, m/s
5  DRONE_FIXED_HEIGHT = 1800  # 无人机固定高度, m
6  M1_INITIAL = np.array([20000, 0, 2000])  # M1初始位置
7  FY1_INITIAL = np.array([17800, 0, DRONE_FIXED_HEIGHT])  # FY1初始位置（高度固定）
8  GRAVITY = 9.8  # 重力加速度, m/s²
9
10
11 # 1. 位置计算函数
```

```python
def missile_pos(t):
    """计算导弹M1在时间t的位置"""
    direction = np.array([-M1_INITIAL[0], -M1_INITIAL[1], -M1_INITIAL[2]])
    unit_direction = direction / np.linalg.norm(direction)
    return M1_INITIAL + MISSILE_SPEED * unit_direction * t


def drone_pos(t, phi, drone_speed, t_launch):
    """计算无人机在时间t的位置 (保持1800m高度) """
    if t < 0:
        return FY1_INITIAL
    dir_x = np.cos(phi)
    dir_y = np.sin(phi)
    dir_z = 0  # 保持高度
    flight_time = min(t, t_launch) if t_launch > 0 else t
    return FY1_INITIAL + drone_speed * np.array([dir_x, dir_y, dir_z]) *
    flight_time


def smoke_center(t, phi, drone_speed, t_launch, t_delay):
    """计算烟幕中心在时间t的位置"""
    t_detonate = t_launch + t_delay
    launch_pos = drone_pos(t_launch, phi, drone_speed, t_launch)

    if t < t_launch:  # 烟雾弹尚未投放
        return None

    if t < t_detonate:  # 平抛运动阶段
        fall_time = t - t_launch
        dir_x = np.cos(phi)
        dir_y = np.sin(phi)
        x = launch_pos[0] + drone_speed * dir_x * fall_time
        y = launch_pos[1] + drone_speed * dir_y * fall_time
        z = launch_pos[2] - 0.5 * GRAVITY * fall_time ** 2
        z = max(z, 0)
        return np.array([x, y, z])
    else:  # 烟幕扩散阶段
        fall_time = t_detonate - t_launch
        dir_x = np.cos(phi)
        dir_y = np.sin(phi)
        x = launch_pos[0] + drone_speed * dir_x * fall_time
        y = launch_pos[1] + drone_speed * dir_y * fall_time
        z = launch_pos[2] - 0.5 * GRAVITY * fall_time ** 2
        z = max(z - 3 * (t - t_detonate), 0)
        return np.array([x, y, z])


```

```python
# 2. 角度计算函数
def cone_axis(t, phi, drone_speed, t_launch, t_delay):
    """计算锥体轴线向量"""
    sc = smoke_center(t, phi, drone_speed, t_launch, t_delay)
    if sc is None:
        return None
    mp = missile_pos(t)
    return sc - mp


def cone_semi_angle(t, phi, drone_speed, t_launch, t_delay):
    """计算投影锥体的半顶角"""
    axis = cone_axis(t, phi, drone_speed, t_launch, t_delay)
    if axis is None:
        return None
    ms_norm = np.linalg.norm(axis)
    return np.arcsin(10 / ms_norm) if ms_norm > 10 else np.pi / 2


def cylinder_point(phi_cyl, gamma):
    """生成圆柱体上的点"""
    alpha = 7 * np.cos(phi_cyl)
    beta = 200 + 7 * np.sin(phi_cyl)
    return np.array([alpha, beta, gamma])


def max_angle_OPY(t, phi, drone_speed, t_launch, t_delay, n_samples=30):
    """找到圆柱体上使角OPY最大的点"""
    sc = smoke_center(t, phi, drone_speed, t_launch, t_delay)
    if sc is None:
        return 0  # 尚未投放烟雾弹，无遮蔽

    mp = missile_pos(t)
    max_angle = 0
    phi_samples = np.linspace(0, 2 * np.pi, n_samples)
    gamma_samples = np.array([0, 10])  # 增加中间采样点

    for phi_cyl in phi_samples:
        for gamma in gamma_samples:
            Y = cylinder_point(phi_cyl, gamma)
            PO = sc - mp
            PY = Y - mp
            dot_product = np.dot(PO, PY)
            norm_PO = np.linalg.norm(PO)
            norm_PY = np.linalg.norm(PY)
            if norm_PO == 0 or norm_PY == 0:
                continue
```

```
105             cos_angle = dot_product / (norm_PO * norm_PY)
106             angle = np.arccos(np.clip(cos_angle, -1.0, 1.0))
107             if angle > max_angle:
108                 max_angle = angle
109     return max_angle
110
111
112 # 3. 有效遮蔽判断
113 def is_effective(t, phi, drone_speed, t_launch, t_delay):
114     """判断在时间t是否有效遮蔽"""
115     sc = smoke_center(t, phi, drone_speed, t_launch, t_delay)
116     if sc is None:
117         return False
118
119     max_angle = max_angle_OPY(t, phi, drone_speed, t_launch, t_delay)
120     semi_angle = cone_semi_angle(t, phi, drone_speed, t_launch, t_delay)
121
122     if max_angle is None or semi_angle is None:
123         return False
124
125     return max_angle < semi_angle
126
127
128 # 4. 粒子群优化算法
129 class Particle:
130     def __init__(self, bounds):
131         self.position = np.array([np.random.uniform(low, high) for low, high in
        bounds])
132         self.velocity = np.array([np.random.uniform(-1, 1) for _ in bounds])
133         self.best_position = self.position.copy()
134         self.best_score = -np.inf
135
136
137 def pso_optimize(objective_func, bounds, num_particles=30, max_iter=50, w=0.5, c1
        =1, c2=2):
138     particles = [Particle(bounds) for _ in range(num_particles)]
139     global_best_position = particles[0].position.copy()
140     global_best_score = -np.inf
141
142     for particle in particles:
143         score = objective_func(particle.position)
144         if score > particle.best_score:
145             particle.best_score = score
146             particle.best_position = particle.position.copy()
147         if score > global_best_score:
148             global_best_score = score
149             global_best_position = particle.position.copy()
```

```
150
151     for iter in range(max_iter):
152         for particle in particles:
153             r1, r2 = np.random.random(2)
154             cognitive = c1 * r1 * (particle.best_position - particle.position)
155             social = c2 * r2 * (global_best_position - particle.position)
156             particle.velocity = w * particle.velocity + cognitive + social
157
158             particle.position += particle.velocity
159             for i in range(len(bounds)):
160                 if particle.position[i] < bounds[i][0]:
161                     particle.position[i] = bounds[i][0]
162                     particle.velocity[i] = -particle.velocity[i] * 0.5
163                 elif particle.position[i] > bounds[i][1]:
164                     particle.position[i] = bounds[i][1]
165                     particle.velocity[i] = -particle.velocity[i] * 0.5
166
167             score = objective_func(particle.position)
168             if score > particle.best_score:
169                 particle.best_score = score
170                 particle.best_position = particle.position.copy()
171             if score > global_best_score:
172                 global_best_score = score
173                 global_best_position = particle.position.copy()
174
175         if (iter + 1) % 10 == 0:
176             print(f"迭代 {iter + 1}/{max_iter}, 当前最优分数: {global_best_score:.2
    f}s")
177
178     return global_best_position, global_best_score
179
180
181 # 5. 目标函数
182 # def objective_function(params):
183 #     """目标函数: 计算有效遮蔽时间"""
184 #     phi, drone_speed, t_launch, t_delay = params
185 #
186 #     # 计算关键时间点
187 #     detonate_time = t_launch + t_delay
188 #     end_time = detonate_time + 20
189 #
190 #     # 导弹飞行总时间
191 #     missile_total_time = np.linalg.norm(M1_INITIAL) / MISSILE_SPEED
192 #     if end_time > missile_total_time:
193 #         end_time = missile_total_time
194 #
195 #     # 检查时间段
```

```
196 #     start_check = max(detonate_time, 0)  # 确保检查时间不早于起爆时间
197 #     if start_check >= end_time:
198 #         return 0
199 #
200 #     step = 0.2  # 时间步长
201 #     t = start_check
202 #     effective_time = 0
203 #
204 #     while t <= end_time:
205 #         if is_effective(t, phi, drone_speed, t_launch, t_delay):
206 #             effective_time += step
207 #         t += step
208 #
209 #     return effective_time
210 def objective_function(params):
211     """变步长精确计算有效遮蔽时间"""
212     phi, drone_speed, t_launch, t_delay = params
213
214     detonate_time = t_launch + t_delay
215     missile_total_time = np.linalg.norm(M1_INITIAL) / MISSILE_SPEED
216     start_check = max(detonate_time, 0)
217     end_time = min(detonate_time + 20, missile_total_time)
218     if start_check >= end_time:
219         return 0.0
220
221     # ---- 第1遍: 0.5 s 大步长找区间 ----
222     coarse_step = 0.5
223     coarse_intervals = []
224     cur_start = None
225     t = start_check
226     while t <= end_time:
227         if is_effective(t, phi, drone_speed, t_launch, t_delay):
228             if cur_start is None:
229                 cur_start = t
230         else:
231             if cur_start is not None:
232                 coarse_intervals.append((cur_start, t))
233                 cur_start = None
234         t += coarse_step
235     if cur_start is not None:
236         coarse_intervals.append((cur_start, end_time))
237
238     # ---- 第2遍: 0.01 s 精细扫描 ----
239     fine_step = 0.001
240     effective_time = 0.0
241     for s, e in coarse_intervals:
242         t = s
```

```
243             while t < e:
244                 if is_effective(t, phi, drone_speed, t_launch, t_delay):
245                     effective_time += fine_step
246                 t += fine_step
247         return effective_time
248 
249 # 6. 主程序
250 if __name__ == "__main__":
251         bounds = [
252             (0, np.pi/2),  # phi
253             (70, 140),  # drone_speed
254             (0, 30),  # t_launch
255             (0, 50)  # t_delay
256         ]
257 
258         print("开始粒子群优化...")
259         best_params, best_score = pso_optimize(
260             objective_function,
261             bounds,
262             num_particles=20,
263             max_iter=100,
264             w=0.7,
265             c1=1.5,
266             c2=1.5
267         )
268 
269         phi_opt, drone_speed_opt, t_launch_opt, t_delay_opt = best_params
270         phi_deg = np.degrees(phi_opt)
271 
272         dir_x = np.cos(phi_opt)
273         dir_y = np.sin(phi_opt)
274         launch_position = drone_pos(t_launch_opt, phi_opt, drone_speed_opt,
            t_launch_opt)
275         detonate_time = t_launch_opt + t_delay_opt
276         detonate_position = smoke_center(detonate_time, phi_opt, drone_speed_opt,
            t_launch_opt, t_delay_opt)
277 
278         print("\n优化结果:")
279         print(f"最大有效遮蔽时间: {best_score:.2f}秒")
280         print(f"无人机最优飞行速度: {drone_speed_opt:.2f} m/s")
281         print(f"无人机飞行方向:")
282         print(f"  方位角(phi): {phi_deg:.2f}°")
283         print(f"  方向向量: ({dir_x:.4f}, {dir_y:.4f}, 0.0000)")
284         print(f"最佳投放时间: {t_launch_opt:.2f}秒")
285         print(f"投放点位置: ({launch_position[0]:.2f}, {launch_position[1]:.2f}, {
            launch_position[2]:.2f})")
286         print(f"最佳起爆延迟: {t_delay_opt:.2f}秒")
```

```
287         print(f"实际起爆时间: {detonate_time:.2f}秒")
288         print(f"起爆点位置: ({detonate_position[0]:.2f}, {detonate_position[1]:.2f}, {
        detonate_position[2]:.2f})")
```

Listing 2: 问题 2 求解代码