无人机侦听与反制中的规划与决策问题

摘要

旋翼无人机由于其兼具灵活、低廉、适应性强的优点，来被广泛应用与战场的侦察、打击中；而对无人机进行防御也是一个重要的问题，最佳的防御是主动进攻，如果能在无人机抵近目标之间就将其迫降、截获，既能获得最高的安全保障，又能获得对方的技术情报。

无人机对通信基站进行抵近侦察时，需要综合考虑基站数量、无人机与基站距离、无人机与基站的相对位置、无人机的飞行速度等问题，本文提出了一种尽快进入其中一个基站的被侦听区域，根据无人机测量得到的侦听所有基站所接受的和信息速率评价数据，做出飞行方向的决策，使得平均和信息速率尽快收敛到较高水平。

使用干扰机对旋翼无人机进行反制时，主要基于对无人机飞控信号的干扰，尽快测量得到无人机飞控信号的跳频周期和跳频规律，进而调整发送频率变化规律，使得无人机飞控信号在一个周期内所有飞控信号都不能被争取译码，发出 sos 信号执行降落，完成迫降无人机的任务。本文提出了跳频规律测量算法以及跳频周期测量算法两种算法，能高效得反置无人机。

在抵近侦察和干扰反制方案互相攻防的基础上，本文对干扰机的配置（位置、功率、数量）进行模拟权衡，以尽可能少得干扰机，尽可能低的功率，尽快迫降无人机，使得无人机无法进入任何一个基站的被侦听区域，并给出了仿真航迹图和 50 次实验的无人机迫降位置。

关键字： 旋翼无人机　抵近侦听　无人机反制

一、 问题重述

旋翼无人机侦听：旋翼无人机从起始位置 (a_0, b_0) 出发，在恒定高度 H = 150 米巡航，对已知数量 K 个通信基站（其位置为 $(a_1, b_1), ..., (a_K, b_K)$，其高度忽略不计）进行侦听，但是无人机并不知晓通信基站的确切位置，因此旋翼无人机持续搜索以发现最佳的侦听位置，即通过决策前、后、左、右搜索方向，最大化长期折扣预期。

旋翼无人机的反制：旋翼无人机飞行中需保证飞控信号正常接收，飞控信号以跳频形式传输，即在不同时刻选择不同频率传输，频率变换每秒发生 217 次。若在飞控信号占据的时-频资源块上出现干扰信号，且干扰信号功率与噪声功率 σ^2 的比值高于门限 $\rho_2 = 3.1623$（约为 5 dB）时，则无人机不能译码相应时-频资源块的飞控信号，无人机发出 NACK（重传请求）信号，要求重传飞控信号；如果一个周期的飞控信号都被干扰不能译码，无人机发出 sos 信号，并自行降落。干扰机与无人机之间也存在信道功率增益，即干扰信号在传播过程中会衰减。

问题 1 给定旋翼无人机的起始水平坐标 $(100, 100)$，旋翼无人机计划侦听 3 个通信基站，坐标分别为 $(1350, 1950)$、$(1600, 1790)$、$(1590, 2090)$。当信噪比 $\frac{Ph_k}{\sigma^2}$ 小于门限 $\rho_1 = 3.1623$（约为 5 dB）时，信息不能正确译码。建立包含旋翼无人机、通信基站、干扰机等的位置、航迹、信道、信号模型，对路径决策与规划问题建模，并推导求解问题的方法，在此基础上建立旋翼无人机完成侦听任务的航迹决策与规划算法，并解释算法参数选择。生成无人机搜索侦听的航迹图；结合建立的模型，指出旋翼无人机首次进入任一通信基站可被侦听区域的时间；研究平均和信息速率评价函数与时间的关系。

问题 2 研究旋翼无人机反制，其中旋翼无人机实际飞控信号跳频图样如图 2 所示。假设干扰机发送信号到达旋翼无人机时的信号功率满足门限 ρ2 ，若干扰机已知无人机飞控信号跳频频率以及跳频周期，但不知具体跳频图样，试设计算法，使旋翼无人机迫降。如果干扰机未知跳频周期，仅知道每个跳频时刻飞控信号只在一个频率传输，试设计算法迫降无人机。注意：说明算法参数取值选择，并讨论算法的收敛速度，并指出 20 次仿真时，平均迫降无人机的时间（注：跳频时频率变换每秒发生 217 次）。

问题 3 在问题 1 和问题 2 的基础上，假设旋翼无人机的水平飞行速度 0≤v0≤ 10 米/秒，干扰机未知跳频周期，仅知道每个跳频时刻飞控信号只在一个频率传输（如图 2 所示）。3 个基站的坐标分别为 (1350, 1950)、(1600, 1790)、(1590, 2090)。旋翼无人机随机从 (100, 100) 和 (2700, 3800) 两个位置出发。研究 50 次仿真中，基站均不被侦听时，求解干扰机的功率、部署位置、部署数量。即以尽可能少的干扰机，尽可能低的发射功率，尽可能

靠近基站处部署，在尽可能远处迫降旋翼无人机（此时，无人机须迫降在基站可被侦听的区域的边界外，即确保基站不被侦听）。需用仿真结果/图（如航迹、无人机位置等）说明以上要求均被满足。

二、 模型假设

1. 干扰机收到的 NACK 信号的传播耗时忽略不计。
2. 基站集中于某个区域，且基站的侦听范围有交集的区域，即最佳侦听范围。
3. 无人机返回的 NACK 信号为未正常译码的对应的跳频信号频率，干扰机可以收到并解析出其频率。
4. 假设干扰机体积不占空间

三、 符号说明

符号	意义
a_0	无人机起始位置横坐标
b_0	无人机起始位置纵坐标
H	无人机恒定高度，值为 150 米
K	通信基站的个数
$x(t)$	无人机在 t 时刻位置的横坐标
$y(t)$	无人机在 t 时刻位置的纵坐标
$d_k(t)$	无人机在 t 时刻与第 k 个通信基站之间的距离
R_k	无人机收到第 k 个通信基站的信息速率
P	基站的发送功率
h_k	第 k 个基站与无人机之间的信道功率增益，满足 $h_k = \frac{1}{d_k^2(t)}$
σ^2	噪声功率，值为 2×10^{-5} 瓦特
γ	折扣因子，满足 $0 < \gamma < 1$，假设为 0.9
ρ_1	信噪比门限
ρ_2	干扰信号功率与噪声功率的比值门限
v_0	无人机水平飞行速度
r	基站的的侦听半径
R	干扰机的有效干扰范围
n	干扰机的数量
P'	干扰机的功率
α	干扰半径和侦听半径的比值
$t_{interrupt}$	干扰机已知跳频周期时，发现飞控信号规律并迫降无人机所需时间
x_i	干扰机位置的横坐标，$i \in [1, n]$
y_i	干扰机位置的纵坐标，$i \in [1, n]$

四、 问题分析

4.1 对问题 1 的回答

根据题目条件可知，在 t 时刻旋翼无人机 [2] 的位置为 $(x(t), y(t))$，则此时无人机与第 k 个通信基站之间的距离为：

$$d_k(t) = \sqrt{H^2 + (x(t) - a_k)^2 + (y(t) - b_k)^2}$$

结合已知条件 $h_k = \frac{1}{d_k^2(t)}$，进一步可得，无人机收到第 k 个通信基站的信息速率评价函数：

$$R_k = \log_2(1 + \frac{Ph_k}{\sigma^2}) = \log_2(1 + \frac{P}{d_k^2(t)\sigma^2})$$

由于当信噪比 $\frac{Ph_k}{\sigma^2}$ 小于门限 $\rho_1 = 3.1623$（约为 5 dB）时，信息不能正确译码，可以计算得到每个基站的侦听半径 r 的值为：

$$r = \sqrt{\frac{P}{\rho_1 \sigma^2}} \approx 397.6m$$

无人机侦听 K 个通信基站所接受的和信息速率评价函数为：

$$R(t) = \sum_{k=1}^{K} R_k$$

对于无人机未知的基站坐标 $(1350, 1950)$、$(1600, 1790)$、$(1590, 2090)$，我们预先绘制出一定区域内以 $R(t)$ 的值为竖坐标的曲面图：

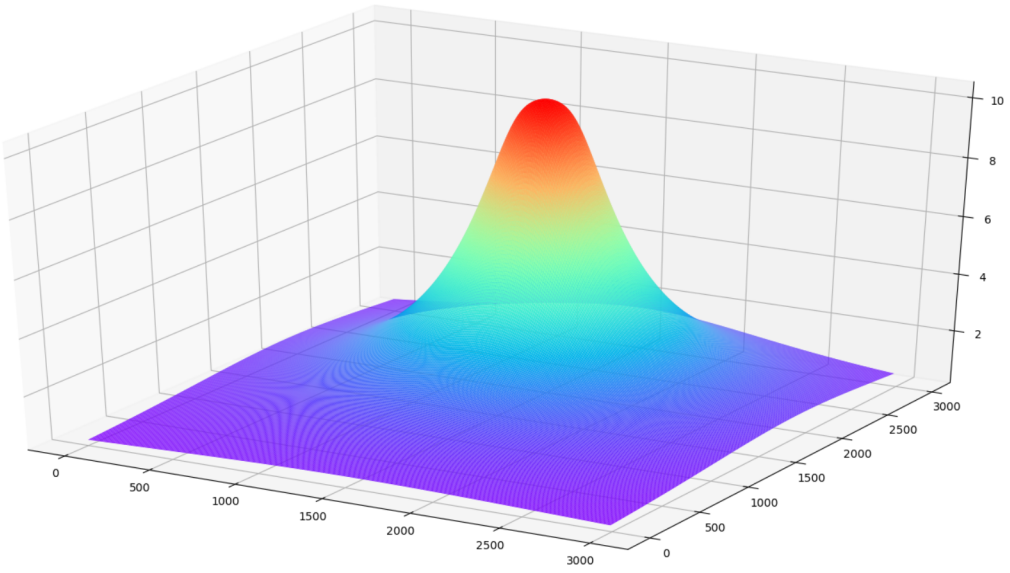

图 **1**　以 $R(t)$ 为竖坐标的曲面图

可以看到，三个基站共同形成的 $R(t)$ 值存在一个峰值区域，而这个峰值区域必然就是共同的被侦听区域。

无人机抵近侦听算法　为了尽快进入侦听半径，将无人机的速度 v_0 设置为 10m/s，令飞行周期 T=1s，无人机不断地测量 $R(t)$ 的值，并根据 $R(t)$ 值的变化决策下一步的移动方向。初始时默认按照一周期前进一周期右移的规律交替移动，若 $R(t)$ 的值减小，说明在这个方向上已经收敛，则不再向这个方向移动，当横纵两个方向都已经收敛时无人机一定已经进入三个基站信号叠加的峰值附近，已经可以成功译码信息。

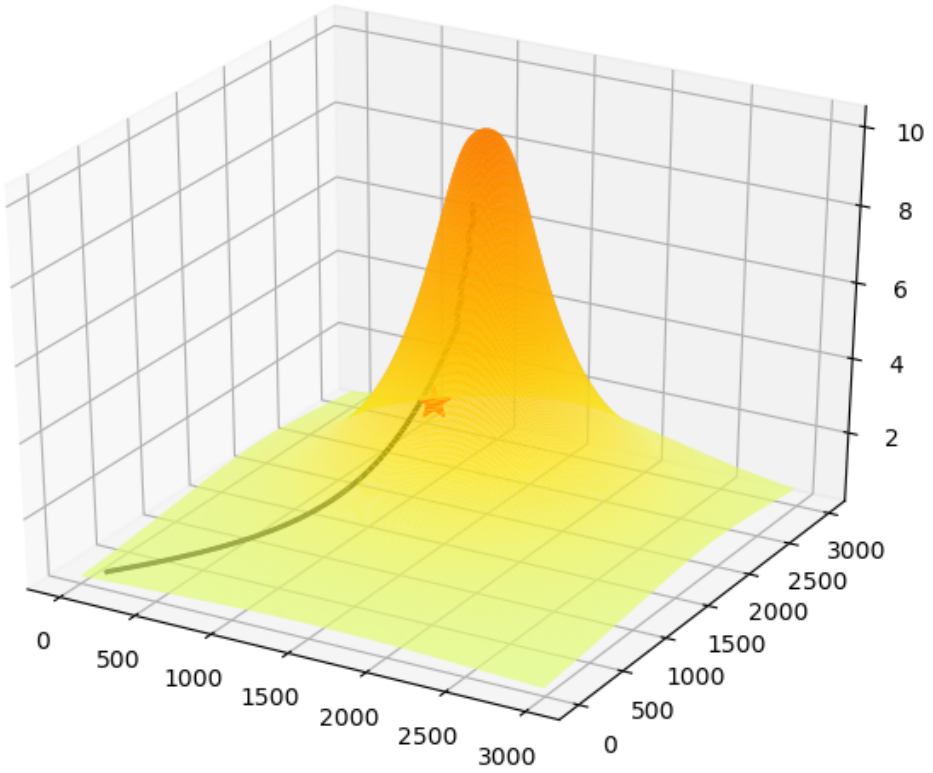

图 **2**　问题一：无人机飞行路径（星号表示无人机进入某个基站的侦听范围内）

根据我们的实验效果，若速度固定为 10m/s，未出现一个飞控信号周期的飞控信号都无法解码而迫降的情况时，326 秒能够到达三个基站共同的可被侦听区域内的点 $(1550, 1730)$，且在第 271 秒时第一次到达一个基站的可被侦听区域内的点 $(1460, 1450)$。这个过程最大化长期折扣预期 $maxE\{\sum_{t=1}^{\infty}\gamma^{t-1}R(t)\} \approx 3.9533$。

4.2 对问题 2 的回答

4.2.1 已知无人机飞控信号跳频频率以及跳频周期

跳频规律测量算法　已知无人机的跳频频率为 $\frac{1}{217}s$，跳频周期为 $\frac{6}{217}s$，使用穷举法，只需要 $\frac{6\times6}{217}s \approx 0.1659s$ 就可以得到无人机的跳频规律。首先将干扰机 [1] 的频率全部为 f_1，第一个周期即可发现 f_1 对应的时间（以干扰机第一次收到对应时间段的 NACK 信号为准），然后除了 f_1 外，其他时间的频率均换为 f_2，直到收到 NACK 以确定 f_2 对应的周

期内时间段，然后固定 f_2，以此类推，共耗时 $\frac{36}{217}s$。接下来干扰机按测量得到的跳频信号变化规律调整干扰信号的频率，对计飞控信号进行干扰，即无人机一个周期的飞控信号都被干扰不能译码，发出 sos 信号，并自行降落，用时 $\frac{6}{217}s$。整个寻找飞控信号规律并迫降无人机共耗时 $\frac{36+6}{217} \approx 0.1935s$，具体的算法以及代码实现见附录 B。

4.2.2 已知无人机飞控信号跳频频率，未知跳频周期

假设设置干扰机数量 $n = 3$，位置均为三个通信基站的外心，分别记为 (x_1, y_1)、(x_2, y_2)、(x_3, y_3)，则干扰机的位置可由下式得到（其中，$i \in [1, n]$）：

$$\begin{cases} x_i = \frac{a_1+a_2+a_3}{3} \\ y_i = \frac{b_1+b_2+b_3}{3} \end{cases}$$

跳频周期测量算法　对于未知调频周期的情况，我们首先设计了一个算法探测跳频周期。算法为三个干扰机在每 $\frac{1}{217}s$ 内随机设置三个不同的频率，若该干扰机收到了 NACK 则不再改变，直到再次收到 NACK，计算两次收到 NACK 的时间差，得到初步的周期测量结果。直到三个干扰机的推断周期均保持稳定，则可以确定无人机的跳频周期（即使测量结果不相同，可能是某个频率在一周期内出现多次导致的，则取三个测量结果的最小公倍数），具体算法即代码实现见附录 C。

得到跳频周期以后，就可以使用上文提出的跳频规律测量算法进行跳频规律的测量和迫降无人机。我们进行了 20 次仿真实验，测量得到的结果平均值约为 0.2933s。

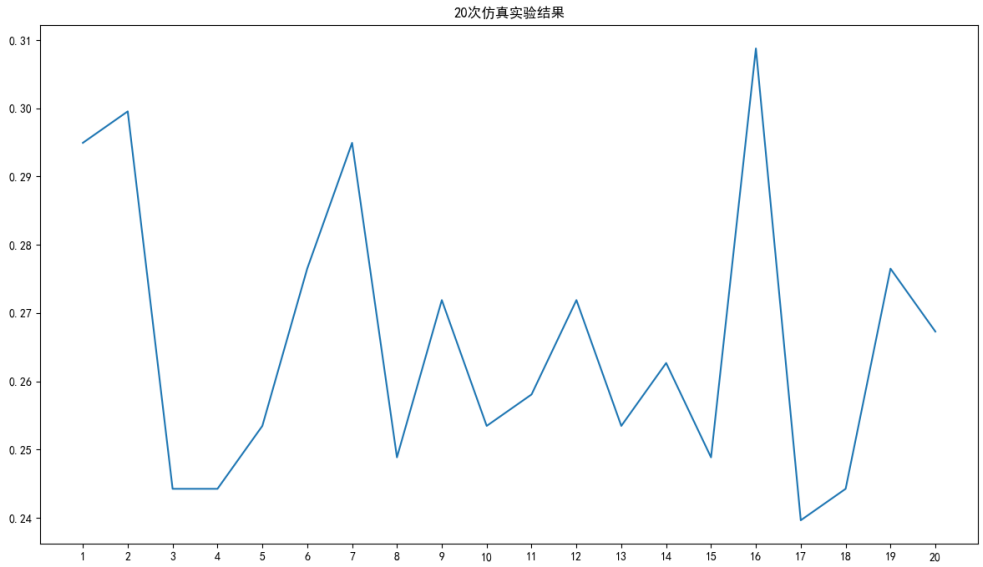

图 **3**　未知跳频周期条件下的 **20** 次仿真实验结果

4.3 对问题 3 的回答

干扰机的功率和其有效干扰范围有关，由于干扰机布置在外心上，故干扰机的干扰范围半径应为干扰机与基站距离、基站被侦听区域半径、极限时间（极限时间内无人机就可以进入被侦听区域）的和：

$$R = \sqrt{H^2 + ((x_i - a_k)^2 + (y_i - b_k)^2)} + r + v_0 \times t_{interrupt}$$

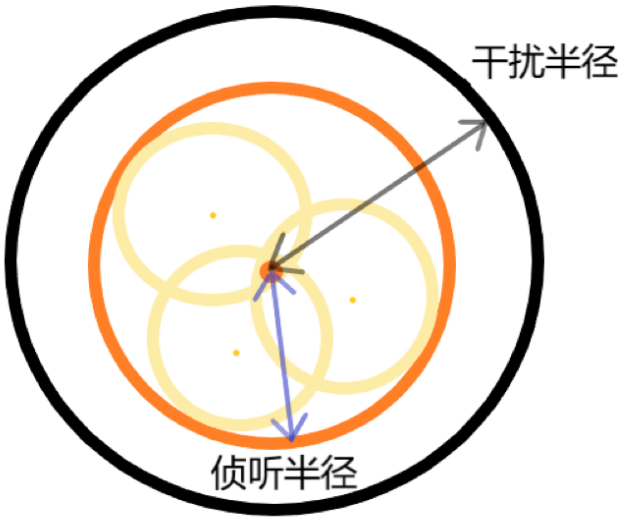

图 4　干扰机有效侦听半径求解

则干扰半径和侦听半径的比值 $\alpha = \frac{R}{r}$。由信噪比的定义可得：

$$\frac{P}{r^2\sigma^2} = \frac{P'}{R^2\sigma^2}$$

故可以得到干扰机功率得表达式：

$$P' = \frac{PR^2}{r^2} = P\alpha^2 \approx 24.636W$$

由于无人机已知干扰机得跳频频率，不妨令 $T = \frac{1}{217}s$，能够减小一部分冗余的移动。假设无人机根据上文提出的无人机抵近侦听算法进行移动，布置两台干扰机于三个基站的外心 $(1513.333, 1943.333)$ 处，使用上文提出的跳频规律测量算法和跳频周期测量算法。

由实验结果可以看出，无人机一旦进入干扰范围，还未进入任何一个通信基站的被侦听范围，就会因为一个跳频周期的飞控信号都被干扰不能译码，发出 sos 信号，自动迫降，干扰成功。

本团队对此场景进行了 50 次仿真实验，如图 7，很显然，所有无人机迫降的位置和三个通信基站之间的距离都大于最大侦听半径，即迫降点均在被侦听区域外。

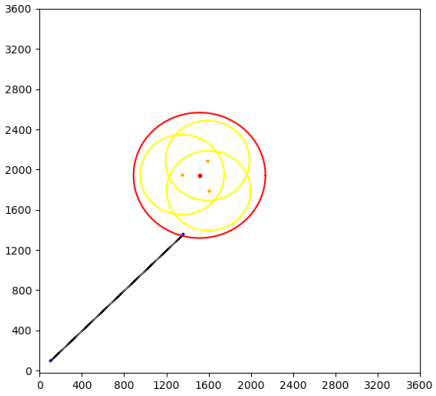

图 **5**　无人机从 $(100, 100)$ 出发

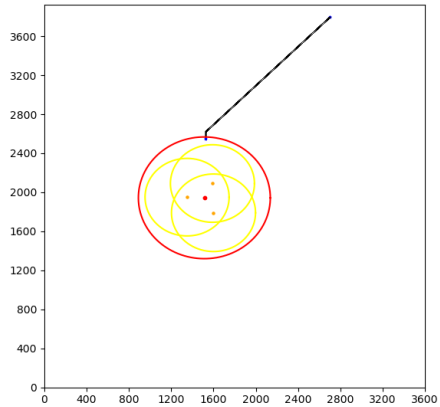

图 **6**　无人机从 $(2700, 3800)$ 出发

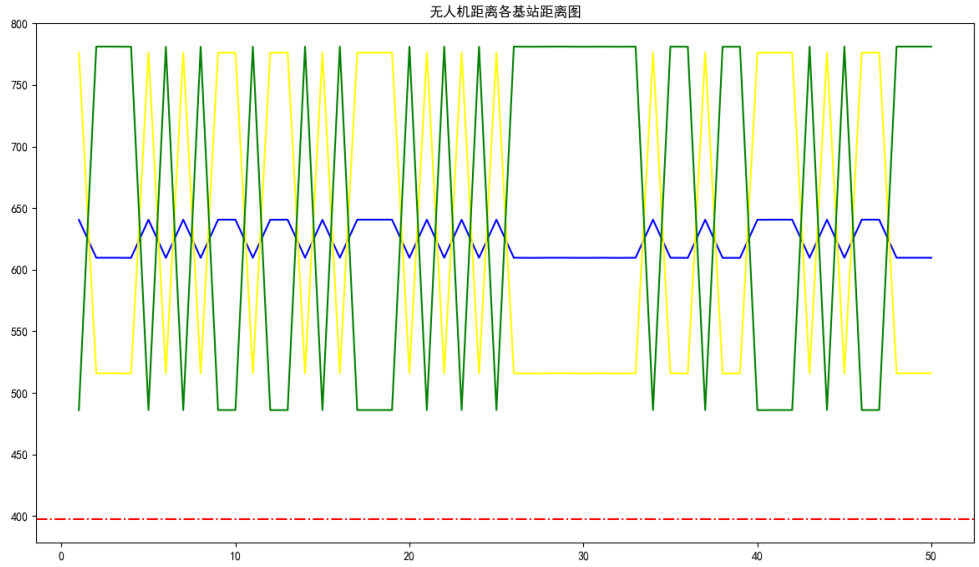

图 **7**　**50** 次仿真实验（红色虚线为最大侦听半径）

五、 总结

本文以旋翼无人机抵近侦听通信基站为背景，分别从无人机抵近侦听的角度和干扰机防御迫降无人机的角度出发，进行攻防，建立了一套在现有条件下尽快抵近侦听的无人机战术，以及一套分为已知无人机跳频周期和未知无人机跳频周期设置的防御算法。

对于问题一，本团队从无人机的飞行特点出发，提出了一种短周期交替飞行，根据无人机检测得到的信息速率评价函数值，动态决策飞行方向的飞行规划算法，实验证明，326 秒即可进入三个基站共同的被侦听区域。

对于问题二，本团队针对已知跳频周期和未知跳频周期的情况分别设计了跳频规

律测量算法以及跳频周期测量算法，实验证明，在已知跳频周期的情况下，只需 0.1935 秒即可测量出无人机的跳频规律，并将其迫降；在未知跳频周期的情况下，平均只需要 0.2923 秒就可以测得跳频周期、跳频规律，并将其迫降。

对于问题三，本团队使用问题一、二中所提出算法，布置干扰机的数量和位置，并对所需功率进行了详细推导计算。本团队进行了 50 次仿真实验，均没有使得无人机进入任何一个通信基站的侦听范围内，证明了算法的有效性。

本文对旋翼无人机抵近通信基站进行侦听、干扰机迫降无人机的场景进行了数学建模，并进行了多次仿真模拟攻防实验，验证了模型的有效性。

参考文献

[1]

[2]

附录 A　问题一：求解无人机飞行线路并作图

```
import math
import random
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# 未知的基站
num = 3
x = [1350, 1600, 1590]
y = [1950, 1790, 2090]

# 求取距离平方
def d2(xx, yy, n):
    return (150**2 + (xx-x[n])**2 + (yy-y[n])**2)

# 求取Rt
def Rt(xx, yy):
    h = (1+10/0.00002/d2(xx, yy, 0))*\
        (1+10/0.00002/d2(xx, yy, 1))*\
        (1+10/0.00002/d2(xx, yy, 2))

    return np.log(h)/np.log(2)

# 求取可侦听的基站数目
def judge(xx, yy, listen_dis):
    listenable = 0;
    for i in range(3):
        if d2(xx, yy, i) <= listen_dis**2:
            listenable += 1
    return listenable

#-----------------------------------#

v = 10 # 恒定速度
t = 0 # 时间变化
t_enter, x_enter, y_enter, enter = 0, 0, 0, False #
gamma = 0.9 # 折扣因子
delta_x, delta_y = 0, 0 # x/y方向上变化后Rt变化值
redium = 397.6 # 最远可侦听到距离
pre_rt = 0 # 上一次Rt
cost = 0 # 折扣预期
_x, _y = 100, 100 # 当前位置
Xtrace, Ytrace = [], []
cost_cumulate = 0 # 折扣预期累加
```

```
max_E = -1 # 最大长期折扣预期
x_direct, y_direct = 1, 1 # 1正半轴方向, -1负半轴方向

#-----------------------------------#

# 判断X/Y方向是否已经收敛, 不收敛则需要继续变动
Xchange = True
Ychange = True
x_limit, y_limit = 10, 10 # 最大反向次数

while Xchange or Ychange:
    # print(_x, _y)
    # X方向

    while Xchange:
        pre_rt = Rt(_x, _y)

        _x += v * x_direct
        Xtrace.append(_x)
        Ytrace.append(_y)
        t += 1

        delta_x = Rt(_x, _y) - pre_rt
        if delta_x < 0:
            x_direct *= -1
            x_limit -= 1
            if not x_limit:
                Xchange = False

        cost += gamma**(t-1) * Rt(_x, _y)
        cost_cumulate += cost
        max_E = max(max_E, cost_cumulate/t)

        if judge(_x, _y, redium)==1 and not enter:
            enter = True
            x_enter = _x
            y_enter = _y
            t_enter = t

        if judge(_x, _y, redium)==3:
            break
        break

    # Y方向

    if not Ychange:
        continue
```

```
        pre_rt = Rt(_x, _y)

        _y += v * y_direct
        Xtrace.append(_x)
        Ytrace.append(_y)
        t += 1

        delta_y = Rt(_x, _y) - pre_rt
        if delta_y < 0:
            y_direct *= -1
            y_limit -= 1
            if not y_limit:
                Ychange = False

        cost += gamma**(t-1) * Rt(_x, _y)
        cost_cumulate += cost
        max_E = max(max_E, cost_cumulate/t)

        if judge(_x, _y, redium)==1 and not enter:
            enter = True
            x_enter = _x
            y_enter = _y
            t_enter = t

        if judge(_x, _y, redium)==3:
            break

    print("First time: (x, y) = (", x_enter, ", ", y_enter, ") T = ", t_enter)
    print("V =", v, " (x, y) = (", _x, ",", _y, ")", " T =", t, " max_E =", max_E)


    #-----------------------------------#

    # 3D展示Rt变化曲线
    fig = plt.figure()
    ax = Axes3D(fig)

    X = np.arange(0, 3000, v)
    Y = np.arange(0, 3000, v)
    X, Y = np.meshgrid(X, Y)
    Z = Rt(X, Y)
    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='Wistia')

    Xtrace = np.array(Xtrace)
    Ytrace = np.array(Ytrace)
    Ztrace = Rt(Xtrace, Ytrace)
    ax.plot(Xtrace, Ytrace, Ztrace, color="black", linewidth=2)
```

```
z_enter = [Rt(x_enter, y_enter)]
x_enter = [x_enter]
y_enter = [y_enter]
# print(x_enter, y_enter, z_enter)
ax.plot(x_enter, y_enter, z_enter, "*",color="r", markersize=15)
plt.show()

#-----------------------------------#

'''
# 速度变化
for v in range(10, 11):
    t = 0 # 时间变化
    gamma = 0.9 # 折扣因子
    delta_x, delta_y = 0, 0 # x/y方向上变化后Rt变化值
    redium = 397.6 # 最远可侦听到距离
    pre_rt = 0 # 上一次Rt
    cost = 0 # 折扣预期
    stride = 5 # 步长
    _x, _y = 100, 100 # 当前位置
    cost_cumulate = 0 # 折扣预期累加
    max_E = -1 # 最大长期折扣预期
    x_direct, y_direct = 1, 1 # 1正半轴方向, -1负半轴方向

    #-----------------------------------#

    # 判断X/Y方向是否已经收敛, 不收敛则需要继续变动
    Xchange = True
    Ychange = True
    x_limit, y_limit = 10, 10 # 最大反向次数

    while True:
        # print(_x, _y)
        # X方向

        for step in range(1, stride+1):
            if not Xchange:
                break
            pre_rt = Rt(_x, _y)

            _x += v * x_direct
            t += 1

            delta_x = Rt(_x, _y) - pre_rt
            if delta_x < 0:
                x_direct *= -1
```

```
                x_limit -= 1
                if x_limit:
                    Xchange = False

            cost += gamma**(t-1) * Rt(_x, _y)
            cost_cumulate += cost
            max_E = max(max_E, cost_cumulate/t)

        if judge(_x, _y, redium)==1:
            stride = 3
        elif judge(_x, _y, redium)==2:
            stride = 1
        elif judge(_x, _y, redium)==3:
            break

        # Y方向

        for step in range(1, stride+1):
            if not Ychange:
                break
            pre_rt = Rt(_x, _y)

            _y += v * y_direct
            t += 1

            delta_y = Rt(_x, _y) - pre_rt
            if delta_y < 0:
                y_direct *= -1
                y_limit -= 1
                if y_limit:
                    Ychange = False

            cost += gamma**(t-1) * Rt(_x, _y)
            cost_cumulate += cost
            max_E = max(max_E, cost_cumulate/t)

        if judge(_x, _y, redium)==1:
            stride = 3
        elif judge(_x, _y, redium)==2:
            stride = 1
        elif judge(_x, _y, redium)==3:
            break


    print("V =", v, " (x, y) = (", _x, ",", _y, ")", " T =", t, " max_E =", max_E)
'''
```

附录 B　问题二：已知无人机飞控信号跳频频率以及跳频周期

```
import math
import random
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D


# 求取距离平方
def d2(xx, yy, n):
    return (150**2 + (xx-x[n])**2 + (yy-y[n])**2)

# 未知的基站信息

num = 3
x = [1350, 1600, 1590]
y = [1950, 1790, 2090]

# 无人机信息
V = 10
T = 6
freq = [5, 1, 4, 2, 3, 6]
redium = 397.6 # 最远可侦听到距离

# 干扰机

P = 10
x_interrupt = np.sum(np.array(x))/3
y_interrupt = np.sum(np.array(y))/3
t_interrupt = 40/217
alpha = (d2(x_interrupt, y_interrupt, 0)**0.5 + redium + t_interrupt * V) / redium
P_interrupt = P * (alpha**2)
freq_interrupt = [1, 1, 1, 1, 1, 1]
freq_found = [False] * 6


print("干扰机坐标 = (", x_interrupt, ",", y_interrupt, ")\n干扰机功率 = ",
     P_interrupt, "瓦特")


Total = 0

for sec217 in range(217*38):
    if all(freq_found):
        Total = sec217/217
```

```
            break
        idx = sec217 % T
        if freq[idx] == freq_interrupt[idx]:
            freq_found[idx] = True
        if idx == T-1:
            for i in range(T):
                if not freq_found[i]:
                    freq_interrupt[i] = freq_interrupt[i] % T + 1
        # print("干扰机频率: ", freq_interrupt, freq_found)

    print("干扰机频率: ", freq_interrupt, "用时: %.12lf"%(Total,))
```

附录 C　问题二：已知无人机飞控信号跳频频率，未知跳频周期

```
import math
import random
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# 未知的基站信息

P = 10 # 基站功率
num = 3 # 基站数目
x = [1350, 1600, 1590]
y = [1950, 1790, 2090]

# 求取距离平方
def d2(xx, yy, n):
    return (150**2 + (xx-x[n])**2 + (yy-y[n])**2)

# 无人机信息
V = 10
T = 6
freq = [5, 1, 4, 2, 3, 6]
'''
T = 18
freq = [3, 4, 3, 4, 3, 4, 5, 6, 1, 5, 3, 1, 5, 1, 4, 2, 3, 6]
'''
redium = 397.6 # 最远可侦听到距离

# 干扰机
t = -1
stability = 0
x_interrupt = np.sum(np.array(x))/3
```

```
y_interrupt = np.sum(np.array(y))/3
t_interrupt = 100/217
n_interrupt = 3
total_time = 0

class Interrupt:

    def __init__(self, order, x, y, d):
        self.order = order
        self.x = x
        self.y = y
        self.dis = d

        self.alpha = ( d2(self.x, self.y, 0)**0.5 + redium + d ) / redium
        self.power = P * (self.alpha**2)
        self.power_cumulate = 0
        self.found = False
        self.freq = -1
        self.freq_span = 6

        self.record = []
        self.T = 0

    def detect(self, x, y):
        return 150**2 + (x-self.x)**2+(y-self.y)**2 <= (redium * self.alpha)**2

    def cal(self):
        power_cumulate += power

    def cal(self, time):
        power_cumulate += power * time

    def freq_random(self):
        if not self.found:
            self.freq = random.randint(1, self.freq_span)


    def freq_process(self, freq_response):
        # 判断是否是自己造成的干扰
        NACK = (freq_response == self.freq)


        # 造成干扰以后确定存在该频率，开始不断记录直至确定周期
        if self.found == True:
            self.record.append(NACK)

        # 匹配
```

```
            if NACK == True:
                # 如果之前已经匹配过，查看是否匹配周期
                if self.found == True:
                    idx = len(self.record)-1
                    if self.record[idx % self.T] != NACK:
                            self.T += 1

                # 如果之前没有匹配过，开始记录并且初始化周期为1
                else:
                    self.found = True
                    self.record.append(NACK)
                    self.T = 1

            # 不匹配
            else:
                # 如果之前已经匹配过，查看是否匹配周期
                if self.found == True:
                    idx = len(self.record)-1
                    if self.record[idx%self.T] != NACK:
                            self.T += 1
                # 如果之前没有匹配过，随便了hh


#------------------------------# 推测周期阶段

IR = [ Interrupt(1, x_interrupt, y_interrupt, t_interrupt * V),\
       Interrupt(2, x_interrupt, y_interrupt, t_interrupt * V),\
       Interrupt(3, x_interrupt, y_interrupt, t_interrupt * V)]

for i, interrupt in enumerate(IR):
    print("No.%d: 坐标 = (%.3lf, %.3lf) 功率 = %.3lf Watt 发出频率 = %d"\
            %(i+1, interrupt.x, interrupt.y, interrupt.power, interrupt.freq))

for sec in range(int(t_interrupt*217)+1):
    f = freq[sec % T]
    print("sec = %d/217, f = %d"%(sec, f))

    for ir in IR:
        ir.freq_random()
    while IR[0].freq == IR[1].freq or\
          IR[1].freq == IR[2].freq or\
          IR[0].freq == IR[2].freq:
        for ir in IR:
            ir.freq_random()

    for ir in IR:
        ir.freq_process(f)
```

```
            print("No. %d:发出频率 = %d, T = %d" % (ir.order, ir.freq, ir.T))

        if IR[0].T != 0 and IR[0].T == IR[1].T and IR[0].T == IR[2].T:
            stability += 1
            if stability >= max(10, IR[0].T):
                t = IR[0].T
                total_time += sec
                break
        else:
            stability = 0

    print("The Interrupters speculate T = ", t)


    #------------------------------# 干扰阶段

    freq_found = [False] * t
    freq_interrupt = [1] * t
    for sec217 in range(6*t+1):
        if all(freq_found):
            print("%.6lf second receive SOS"%((total_time+sec217)/217,))
            # print("%.3f second receive SOS"%(sec217/217,))
            total_time += sec217
            break

        idx = sec217 % T
        if freq[idx] == freq_interrupt[idx]:
            freq_found[idx] = True
            print("%.6lf second receive NACK"%((total_time+sec217)/217,))
        if idx == T-1:
            for i in range(T):
                if not freq_found[i]:
                    freq_interrupt[i] = freq_interrupt[i] % T + 1
        # print("干扰机频率: ", freq_interrupt, freq_found)

    print("干扰机频率: ", freq_interrupt, "\n用时: %.6lf second"%(total_time/217,))
```

附录 D　问题三

```
import math
import random
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import Ellipse, Circle
```

```
# 公共数据
sec = 0 # 时间参数(1/217)
T_found = False # 是否推断出周期
T_defend = False # 是否已经开始仅启动一架干扰机进行防御
SOS = False # 是否成功迫降

#-----------------------------------#

# 基站信息
P = 10 # 基站功率
num = 3 # 基站数目
x = [1350, 1600, 1590]
y = [1950, 1790, 2090]

# 求取到基站的距离平方
def d2(xx, yy, n):
    return (150**2 + (xx-x[n])**2 + (yy-y[n])**2)

#-----------------------------------#

# 无人机信息
_x, _y = [(100, 100), (2700, 3800)][random.randint(0, 1)] # 当前位置
print(_x, _y)
Xtrace, Ytrace = [], []
V = 10
T = 6
freq = [5, 1, 4, 2, 3, 6]
redius = 397.6 # 最远可侦听到距离

delta_x, delta_y = 0, 0 # x/y方向上变化后Rt变化值
pre_rt = 0 # 上一次Rt
x_direct, y_direct = 1, 1 # 1正半轴方向, -1负半轴方向

# 判断X/Y方向是否已经收敛, 不收敛则需要继续变动
Xchange = True
Ychange = True
x_limit, y_limit = 10, 10 # 最大反向次数

# 求取Rt
def Rt(xx, yy):
    h = (1+10/0.00002/d2(xx, yy, 0))*\
        (1+10/0.00002/d2(xx, yy, 1))*\
        (1+10/0.00002/d2(xx, yy, 2))

    return np.log(h)/np.log(2)
```

```
# 求取无人机当前位置(xx, yy)可侦听的基站数目
def judge(xx, yy, listen_dis):
    listenable = 0;
    for i in range(3):
        if d2(xx, yy, i) <= listen_dis**2:
            listenable += 1
    return listenable


#-----------------------------------#

# 干扰机
t_speculate = 0
stability = 0
x_interrupt = np.sum(np.array(x))/3
y_interrupt = np.sum(np.array(y))/3
t_interrupt = 100/217
n_interrupt = 2

freq_found = [False]
freq_interrupt = [1]

class Interrupt:

    def __init__(self, order, x, y, d):
        self.order = order
        self.x = x
        self.y = y
        self.dis = d

        self.alpha = ( d2(self.x, self.y, 0)**0.5 + redius + d ) / redius
        self.power = P * (self.alpha**2)
        self.power_cumulate = 0
        self.found = False
        self.freq = -1
        self.freq_span = 6

        self.record = []
        self.T = 0

    def detect(self, x, y):
        return 150**2 + (x-self.x)**2+(y-self.y)**2 <= (redius * self.alpha)**2

    def cal(self):
        self.power_cumulate += self.power
```

```
    def cal(self, time):
        self.power_cumulate += self.power * time

    def freq_random(self):
        if not self.found:
            self.freq = random.randint(1, self.freq_span)


    def freq_process(self, freq_response):
        # 判断是否是自己造成的干扰
        NACK = (freq_response == self.freq)


        # 造成干扰以后确定存在该频率，开始不断记录直至确定周期
        if self.found == True:
            self.record.append(NACK)

        # 匹配
        if NACK == True:
            # 如果之前已经匹配过，查看是否匹配周期
            if self.found == True:
                idx = len(self.record)-1
                if self.record[idx % self.T] != NACK:
                       self.T += 1

            # 如果之前没有匹配过，开始记录并且初始化周期为1
            else:
                self.found = True
                self.record.append(NACK)
                self.T = 1

        # 不匹配
        else:
            # 如果之前已经匹配过，查看是否匹配周期
            if self.found == True:
                idx = len(self.record)-1
                if self.record[idx%self.T] != NACK:
                       self.T += 1
            # 如果之前没有匹配过，随便了hh

IR = [ Interrupt(1, x_interrupt, y_interrupt, t_interrupt * V),\
      Interrupt(2, x_interrupt, y_interrupt, t_interrupt * V)]

for i, interrupt in enumerate(IR):
    print("No.%d: 坐标 = (%.3lf, %.3lf) 功率 = %.3lf Watt 发出频率 = %d"\
            %(i+1, interrupt.x, interrupt.y, interrupt.power, interrupt.freq))
```

```
#-----------------------------------#

# 仿真阶段

while not SOS:
    # print(_x, _y)
    f = freq[sec % T]

    if IR[0].detect(_x, _y) or IR[1].detect(_x, _y):
        print("sec = %d/217, f = %d"%(sec, f))

    #---------推断/干扰周期--------#

    if IR[0].detect(_x, _y):
        if T_found == True:
            if T_defend == False:
                T_defend = True
                freq_found = freq_found * t_speculate
                freq_interrupt = freq_interrupt * t_speculate
            if all(freq_found):
                print("%.6lf second receive SOS"%((sec+t_speculate)/217,))
                SOS = True
                break

            idx = sec % t_speculate
            if f == freq_interrupt[idx]:
                freq_found[idx] = True
                print("%.6lf second receive NACK"%(sec/217,))
            if idx == t_speculate-1:
                for i in range(t_speculate):
                    if not freq_found[i]:
                        freq_interrupt[i] = freq_interrupt[i] % T + 1


        else:
            while IR[0].freq == IR[1].freq:
                for ir in IR:
                    ir.freq_random()

            for ir in IR:
                ir.freq_process(f)
                print("No. %d:发出频率 = %d, T = %d" % (ir.order, ir.freq, ir.T))
            if IR[0].T != 0 and IR[0].T == IR[1].T:
                stability += 1
                if stability >= max(10, IR[0].T):
                    t_speculate = IR[0].T
                    T_found = True
```

```
            elif IR[0].T != IR[1].T:
                stability = 0

#-------X方向移动-------#

if Xchange:
    pre_rt = Rt(_x, _y)

    _x += V * x_direct / 217
    Xtrace.append(_x)
    Ytrace.append(_y)

    delta_x = Rt(_x, _y) - pre_rt
    if delta_x < 0:
        x_direct *= -1
        x_limit -= 1
        if not x_limit:
            Xchange = False
sec += 1
if T_found:
    IR[0].cal(1/217)
else:
    for ir in IR:
        ir.cal(1/217)

if judge(_x, _y, redius)>=1:
    print("The infrastructure has been listened. Interrupters Failed.")
    break
elif not (Xchange or Ychange):
    print("The plane has stop working. Plane Failed. ")




f = freq[sec % T]

if IR[0].detect(_x, _y) or IR[1].detect(_x, _y):
     print("sec = %d/217, f = %d"%(sec, f))

 #---------推断/干扰周期--------#

if IR[0].detect(_x, _y):
    if T_found == True:
        if T_defend == False:
            T_defend = True
            freq_found = freq_found * t_speculate
            freq_interrupt = freq_interrupt * t_speculate
```

```
                if all(freq_found):
                    print("%.6lf second receive SOS"%((sec+t_speculate)/217,))
                    SOS = True
                    break

                idx = sec % t_speculate
                if f == freq_interrupt[idx]:
                    freq_found[idx] = True
                    print("%.6lf second receive NACK"%(sec/217,))
                if idx == t_speculate-1:
                    for i in range(t_speculate):
                        if not freq_found[i]:
                            freq_interrupt[i] = freq_interrupt[i] % T + 1


            else:
                while IR[0].freq == IR[1].freq:
                    for ir in IR:
                        ir.freq_random()

                for ir in IR:
                    ir.freq_process(f)
                    print("No. %d:发出频率 = %d, T = %d" % (ir.order, ir.freq, ir.T))
                if IR[0].T != 0 and IR[0].T == IR[1].T:
                    stability += 1
                    if stability >= max(10, IR[0].T):
                        t_speculate = IR[0].T
                        T_found = True
                    elif IR[0].T != IR[1].T:
                        stability = 0

        #-------Y方向移动-------#

        if Ychange:
            pre_rt = Rt(_x, _y)

            _y += V * y_direct / 217
            Xtrace.append(_x)
            Ytrace.append(_y)

            delta_y = Rt(_x, _y) - pre_rt
            if delta_y < 0:
                y_direct *= -1
                y_limit -= 1
                if not y_limit:
                    Ychange = False
```

```
        sec += 1
        if T_found:
            IR[0].cal(1/217)
        else:
            for ir in IR:
                ir.cal(1/217)

        if judge(_x, _y, redius)>=1:
            print("The infrastructure has been listened. Interrupters Failed.")
            break
        elif not (Xchange or Ychange):
            print("The plane has stop working. Plane Failed. ")

        continue


print("V=", V, "\n(x, y) = (", _x, ",", _y, ")\n", "Total Time =", sec/217)
print("The speculate T = ", t_speculate, "\nThe frequency is ", freq_interrupt)
for ir in IR:
    print("NO.%d: power consuming = %.6lf"%(ir.order, ir.power_cumulate))

#-----------------------------------#

plt.figure()

plt.scatter(Xtrace[0], Ytrace[0], s=5, marker="*", color="blue")
plt.scatter(Xtrace, Ytrace, s=0.00001, color="black")
plt.scatter(_x, _y, s=5, marker="*", color="blue")

plt.scatter(x, y, s=5, color="orange")
r = redius
for xx, yy in zip(x, y):
    a, b = (xx, yy)
    theta = np.arange(0, 2*np.pi, 0.01)
    xxx = a + r * np.cos(theta)
    yyy = b + r * np.sin(theta)
    plt.plot(xxx, yyy, color="yellow")


plt.scatter(x_interrupt, y_interrupt, s=10, color="red")
r = redius * IR[0].alpha
a, b = (x_interrupt, y_interrupt)
theta = np.arange(0, 2*np.pi, 0.01)
x = a + r * np.cos(theta)
y = b + r * np.sin(theta)
plt.plot(x, y, color="red")
```

```
        my_x_ticks = np.arange(0, 4000, 400)
        my_y_ticks = np.arange(0, 4000, 400)
        plt.xticks(my_x_ticks)
        plt.yticks(my_y_ticks)
        plt.show()

        #-----------------------------------#
```