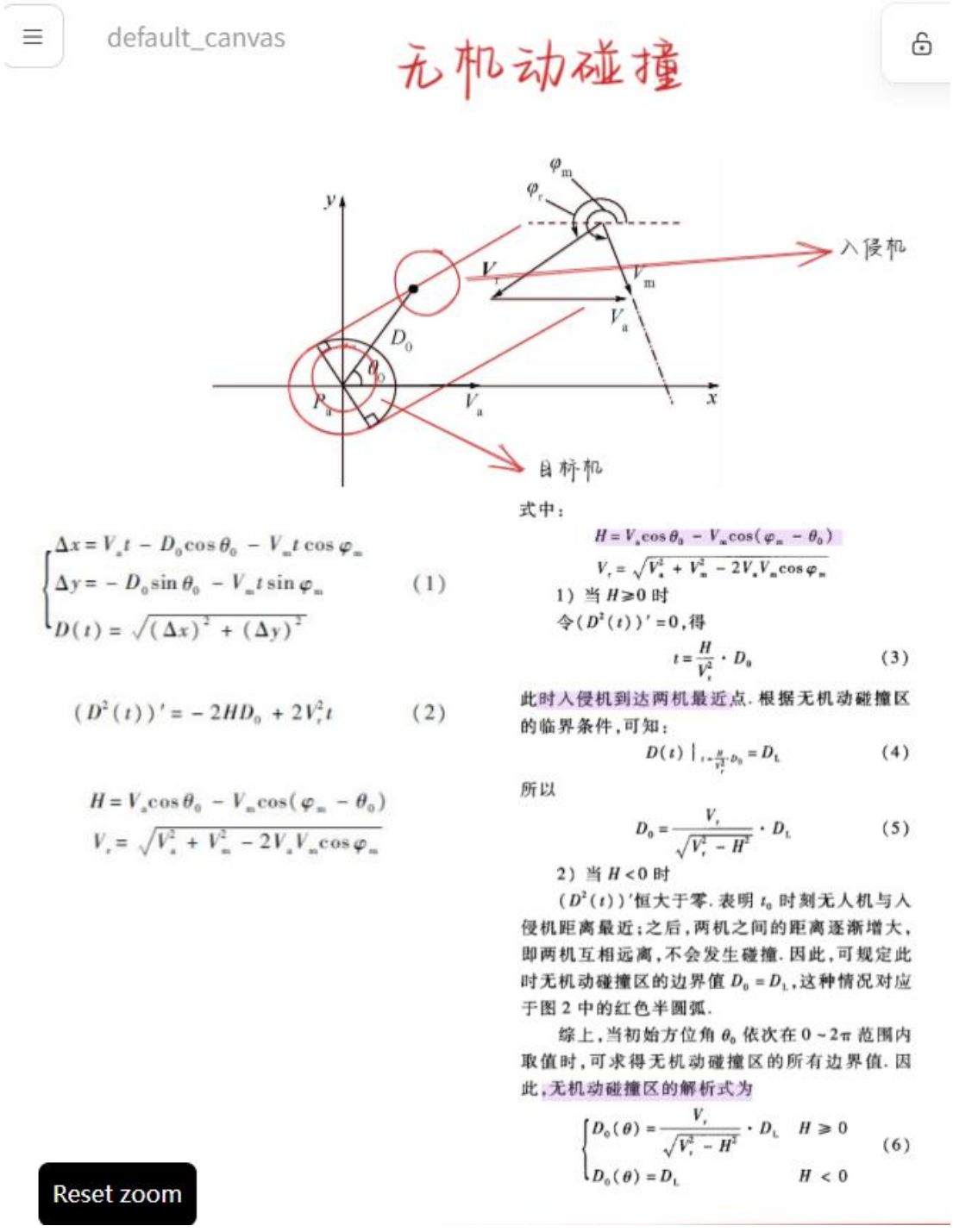
求出某一角度下无机动碰撞区



$$\begin{cases}\Delta x = V_a t - D_0\cos\theta_0 - V_m t\cos\varphi_m \\ \Delta y = -D_0\sin\theta_0 - V_m t\sin\varphi_m \\ D(t) = \sqrt{(\Delta x)^2 + (\Delta y)^2}\end{cases} \quad (1)$$

$$(D^2(t))' = -2HD_0 + 2V_r^2 t \quad (2)$$

$$H = V_a\cos\theta_0 - V_m\cos(\varphi_m - \theta_0)$$
$$V_r = \sqrt{V_a^2 + V_m^2 - 2V_aV_m\cos\varphi_m}$$

式中：

$$H = V_a\cos\theta_0 - V_m\cos(\varphi_m - \theta_0)$$
$$V_r = \sqrt{V_a^2 + V_m^2 - 2V_aV_m\cos\varphi_m}$$

1）当 $H \geqslant 0$ 时

令 $(D^2(t))' = 0$，得

$$t = \frac{H}{V_r^2}\cdot D_0 \quad (3)$$

此时入侵机到达两机最近点. 根据无机动碰撞区的临界条件，可知：

$$D(t)\big|_{t=\frac{H}{V_r^2}\cdot D_0} = D_L \quad (4)$$

所以

$$D_0 = \frac{V_r}{\sqrt{V_r^2 - H^2}}\cdot D_L \quad (5)$$

2）当 $H < 0$ 时

$(D^2(t))'$ 恒大于零. 表明 t_0 时刻无人机与入侵机距离最近；之后，两机之间的距离逐渐增大，即两机互相远离，不会发生碰撞. 因此，可规定此时无机动碰撞区的边界值 $D_0 = D_L$，这种情况对应于图 2 中的红色半圆弧.

综上，当初始方位角 θ_0 依次在 $0 \sim 2\pi$ 范围内取值时，可求得无机动碰撞区的所有边界值. 因此，无机动碰撞区的解析式为

$$\begin{cases}D_0(\theta) = \dfrac{V_r}{\sqrt{V_r^2 - H^2}}\cdot D_L & H \geqslant 0 \\ D_0(\theta) = D_L & H < 0\end{cases} \quad (6)$$

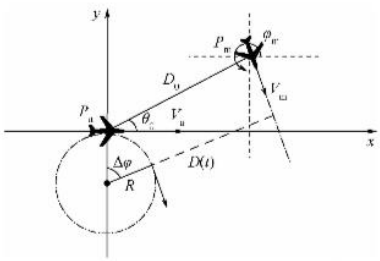
伪代码

```
# 无机动碰撞区计算函数，返回无人机与入侵机的最小安全距离
def calculate_D0(Va, Vm, theta0, phi_m, D_L):
    # 无人机与侵入机的无机动碰撞区
```

```
    H = Va * math.cos(theta0) - Vm * math.cos(phi_m - theta0)  # 无人机和入侵机在初始时刻相对于彼此的速度分量差异
    Vr = math.sqrt(Va**2 + Vm**2 - 2 * Va * Vm * math.cos(phi_m))  # 侵入机相对于无人机的对向速度

    if H >= 0:
        D0_theta = (Vr / math.sqrt(Vr**2 - H**2)) * D_L  # 当 H >= 0 时的 D0(θ)计算
    else:
        D0_theta = D_L  # 当 H < 0 时的 D0(θ)
    return D0_theta
```

求出某一角度下最大机动碰撞区

最大机动碰撞



$$\omega = \frac{d\varphi_a}{dt} = \frac{g}{V_a} n_y \qquad (7)$$

$$R = \frac{V_a}{\omega} = \frac{V_a^2}{g n_y} \qquad (8)$$

$$\begin{cases} \Delta x = R\sin(\omega t) - D_0\cos\theta_0 - V_m t\cos\varphi_m \\ \Delta y = -R + R\cos(\omega t) - D_0\sin\theta_0 - V_m t\sin\varphi_m \\ D(t) = \sqrt{(\Delta x)^2 + (\Delta y)^2} \end{cases} \qquad (9)$$

$$\begin{cases} D'(t) = 0 \\ D(t) = D_L \end{cases} \qquad (10)$$

$$\begin{cases} D_0\left[V_m\cos(\varphi_m - \theta_0) - V_a\cos\left(\frac{gtn_{ymax}}{V_a} + \theta_0\right)\right] + \\ V_m^2 t - V_a V_m t\cos\left(\varphi_m + \frac{gtn_{ymax}}{V_a}\right) + \\ \frac{V_a^3}{gn_{ymax}}\sin\left(\frac{gtn_{ymax}}{V_a}\right) - \\ \frac{V_a^2 V_m}{gn_{ymax}}\sin\left(\varphi_m + \frac{gtn_{ymax}}{V_a}\right) + \frac{V_a^2 V_m}{gn_{ymax}}\sin\varphi_m = 0 \\ (\Delta x)^2 + (\Delta y)^2 - D_L^2 = 0 \end{cases} \qquad (11)$$

伪代码

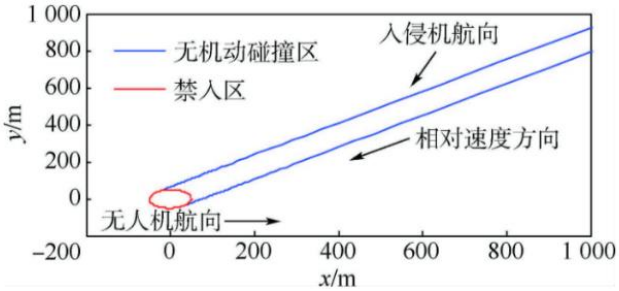
```
def calculate_D0_max(Va, Vm, theta0, phi_m, D_L):
    # 定义初始猜测范围
    t_min = 0  # 时间的最小值
    t_max = 100  # 时间的最大值

    # 定义精度
    epsilon = 1e-6

```

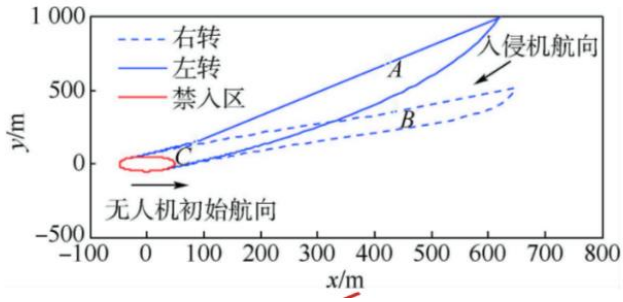
```
    # 定义 f1 函数
    def f1(t, D0):
        term1 = D0 * (Vm * math.cos(phi_m - theta0) - Va * math.cos(g * t * ny_max / Va + theta0))
        term2 = Vm**2 * t - Va * Vm * t * math.cos(phi_m + g * t * ny_max / Va)
        term3 = Va**3 / (g * ny_max) * math.sin(g * t * ny_max / Va)

        term4 = -Va**2 * Vm / (g * ny_max) * math.sin(phi_m + g * t * ny_max / Va)
        term5 = Va**2 * Vm / (g * ny_max) * math.sin(phi_m)
        return term1 + term2 + term3 + term4 + term5

    # 定义 f2 函数
    def f2(t, D0):
        delta_x = Va * t * math.cos(g * t * ny_max / Va) - D0 * math.cos(theta0) - Vm * t * math.cos(phi_m)
        delta_y = Va * t * math.sin(g * t * ny_max / Va) - D0 * math.sin(theta0) - Vm * t * math.sin(phi_m)
        return delta_x**2 + delta_y**2 - D_L**2

    # 定义求解二分法函数
    def bisection_method(f, D0, t_min, t_max, epsilon):
        while (t_max - t_min) > epsilon:
            t_mid = (t_min + t_max) / 2 # 中点计算
            if f(t_mid, D0) * f(t_min, D0) < 0:# 确保两者符号相反
                t_max = t_mid
            else:
                t_min = t_mid
        return (t_min + t_max) / 2

    # 计算 D0 和 t 的值
    D0_initial_guess = D_L
    t_solution = bisection_method(f1, D0_initial_guess, t_min, t_max, epsilon)

    # 微调 D0 使得 f2 接近 0
    def optimize_D0(t_solution, D0_min, D0_max, epsilon):
        while (D0_max - D0_min) > epsilon:
            D0_mid = (D0_min + D0_max) / 2
            if f2(t_solution, D0_mid) * f2(t_solution, D0_min) < 0:
                D0_max = D0_mid
            else:
                D0_min = D0_mid
```

```
        return (D0_min + D0_max) / 2

    D0_solution = optimize_D0(t_solution, 0, D_L * 2, epsilon) # 最大
    return D0_solution
```

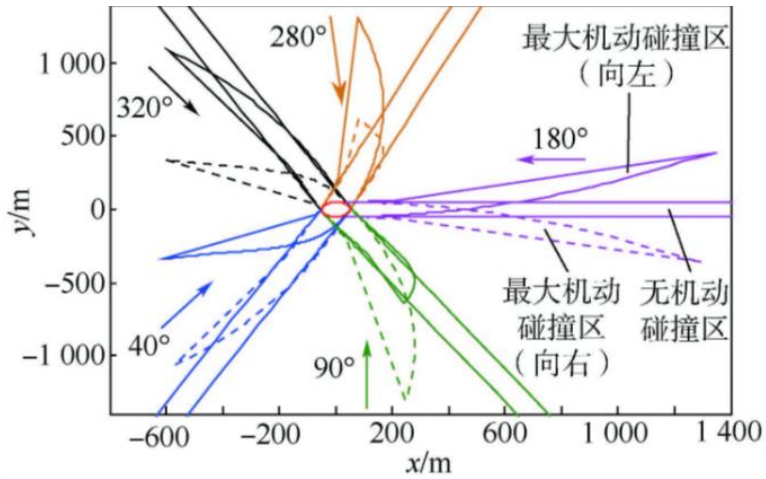
求出拟真条件下两区域边界

```
# 最小安全距离 D =50m
# 无人机飞行速度 =60m/s
# 入侵机飞行速度 Vm= 120m/s
# 无人机航向角 (0)=0,
# 无人机最大侧向过载 n… =0. 8,
# 入侵机航向角 =240。
```





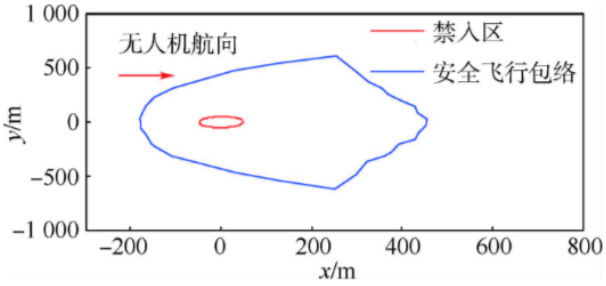
求出不同相对角下的区域边界



由不可规避区求出安全包络

不可规避区：无人机进入该区域，已有规避手段无法进行规避，为最大机动碰撞区与无机动碰撞的交集

安全包络：各个不同的入侵机航向角情况下不可规避区的并集。



各类因素的变化求得不同的安全包络

最小安全距离、无人机速度、入侵机飞行速度、无人及最大侧向过载均会出现不同的包络

可考虑其他不同因素：信号传输时间、时延等因素增加安全距离，完善包络

也可预留出一部分空间时间，来设置警戒区，设置多重包络，增大容错范围