导弹跟踪问题实验报告

王启翔 524072910012

一、引言

在军事领域，导弹跟踪敌艇是一个具有重要战略意义的问题。本实验旨在研究不同情况下导弹对敌艇的追踪过程，通过建立数学模型并运用数值计算方法求解，分析敌艇不同逃逸策略对导弹击中时间和位置的影响，为军事决策和导弹追踪系统设计提供理论依据。

任务 1

某军的一导弹基地发现正北方向 120 km 处海面上有敌艇一艘以 90 km/h 的速度向正东方向方向行驶. 该基地立即发射导弹跟踪追击敌艇，导弹速度为 450 km/h，自动导航系统使导弹在任一时刻都能对准敌艇. 试问导弹在何时何处击中敌艇？（追踪问题） 首先建立如下微分方程组：

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}y} = p & (4.12) \\ \dfrac{\mathrm{d}p}{\mathrm{d}y} = \dfrac{\lambda\sqrt{p^2+1}}{H-y} & (4.13) \\ x(0)=0, p(0)=0 & (4.14) \end{cases}$$

其中，以导弹基地为原点建立平面直角坐标系，正东方向为 x 轴正方向，正北方向为 y 轴正方向。设敌艇速度为v_1，导弹速度为v_2，$\lambda = \dfrac{v_1}{v_2}$，H 为敌艇初始时与导弹基地的 y 方向距离。

应用数学软件 MATLAB 对问题(4.12)~(4.14)进行数值计算，先运用 Euler 进行求解，并输出结果：

MATLAB

```
function e2
% 定义不同的 n 值
j = [4,8,12,24,48,96,120,240];
% 预分配 L 和 T 数组
L = zeros(size(j));T = zeros(size(j));
% 遍历不同的 n 值
for idx = 1:length(j)
n = j(idx);H = 120;h = H/n;lambda = 90/450;x0 = 0;p0 = 0;
```

```
% 进行迭代计算
for i = 0:n-1
x1 = x0 + h*p0;
p1 = p0 + h*(lambda*sqrt(1 + p0^2)/(H - i*h));
x0 = x1;p0 = p1;
end
% 存储每次计算的 L 和 T 值
L(idx) = x1;
T(idx) = x1/90;
end
% 创建表格
resultTable = table(j', L', T', 'VariableNames', {'n', 'L', 'T'});
% 以表格形式显示结果
disp(resultTable);
end
```

Result

```
>> e2
     n        L          T
    ___    ______    _______
      4    11.525    0.12806
      8    15.954    0.17726
     12    17.973     0.1997
     24    20.551    0.22834
     48    22.249    0.24722
     96    23.329    0.25921
    120     23.58      0.262
    240    24.151    0.26834
```

表 4.2

n	4	8	12	24	48	96	120	240
L	11.525	15.964	17.973	20.551	22.249	23.329	23.580	24.151
T	0.1281	0.1773	0.1997	0.2283	0.2472	0.2592	0.2620	0.2683

与表 4.2 的数据比较，发现一致，表明建模正确。

并以更小的步长计算结果，结果如下.

Result

```
>> e2
     n        L          T
    ____    ______    _______
     480    24.497    0.27218
     720     24.63    0.27367
```

1200	24.75	0.275
2040	24.834	0.27593
3600	24.893	0.27659
6000	24.928	0.27698
10000	24.952	0.27724

MATLAB

```
function e3(t, k)
% 设置精度为 10 位
digits(10);
% 定义常量，使用 vpa 转换为可变精度
H = vpa(120);Ve = vpa(90);Vw = vpa(450);t = vpa(t);
% 初始化数组
x = zeros(k + 1, 1, 'sym');y = zeros(k + 1, 1, 'sym');
T = zeros(k + 1, 1, 'sym');
% 初始条件
x(1) = vpa(0);y(1) = vpa(0);T(1) = vpa(0);
% 循环计算每一步的 t, x, y
for i = 1:k
% 计算斜率 M
M = (Ve * T(i) - x(i)) / (H - y(i));
% 更新 x、y 和 T 的值
x(i + 1) = x(i) + Vw * t * M / sqrt(1 + M^2);
y(i + 1) = y(i) + Vw * t / sqrt(1 + M^2);
T(i + 1) = i * t;
% 如果 y 超过 H，则停止循环
if y(i + 1) >= H
k = i; % 更新实际步数
break;
end
end
% 截取实际计算的步数对应的数据
T = T(1:k + 1);
x = x(1:k + 1);
y = y(1:k + 1);
% 创建表格
resultTable = table(T, x, y, 'VariableNames', {'t', 'x', 'y'});
disp(resultTable);
end
```

Result

```
>> e3(0.1,10)
     t           x                y
    ___    ____________    ____________
```

0	0	0
0.1	0	45.0
0.2	5.361534928	89.67945773
0.3	22.67495336	131.2155329

与表 4.2 的数据比较，发现一致，表明建模正确。

表 4.3 $\tau = 0.1$

k	t_k	x_k	y_k
0	0.0	0.000 00	0.000 00
1	0.1	0.000 00	45.000 00
2	0.2	5.361 53	89.679 46
3	0.3	22.674 95	131.215 53

Result

```
>> e3(0.05,10)
      t              x              y
    ____    ___________    ___________

    0          0              0
    0.05       0              22.5
    0.1        1.037357249    44.97607372
    0.15       3.412050748    67.35040792
    0.2        7.646151587    89.44842571
    0.25       14.8678979     110.757963
    0.3        29.19480236    128.1070209
```

再对 Euler 法进行改进，以下为主要改进点：

Euler 法	改进 Euler 法
根据当前点的斜率，直接计算下一个点的近似值，公式为 $y_{n+1} = y_n + h \cdot f(x_n, y_n)$，其中 h 是步长，$f(x_n, y_n)$是在点 (x_n, y_n)处的导数值。这种方法只使用了当前点的信息，误差较大，尤其是在函数变化剧烈时，随着计算步数增加，误差会不断累积。	在这段代码中，改进欧拉法先通过当前点的信息计算出一个预测值（x_star 和 y_star），这类似于简单欧拉法的一步计算。然后，基于预测值再计算一个新的斜率（M2），结合初始斜率（M1） 对预测值进行校正，得到更精确的下一个点的近似值（x(i + 1) 和 y(i + 1)）。这种预测 - 校正机制考虑了更多的信息，能有效减小局部截断误差，提高数值计算的精度。

用改进的 Euler 法计算（步长与 Euler 法相同）

MATLAB

```
function e3(t, k)
```

```
% 设置精度为 20 位
digits(20);
% 定义常量，使用 vpa 转换为可变精度
H = vpa(120);Ve = vpa(90);Vw = vpa(450);t = vpa(t);
% 初始化数组
x = zeros(k + 1, 1, 'sym');y = zeros(k + 1, 1, 'sym');
T = zeros(k + 1, 1, 'sym');
% 初始条件
x(1) = vpa(0);y(1) = vpa(0);T(1) = vpa(0);
% 循环计算每一步的 t, x, y
for i = 1:k
% 避免除零，添加条件判断
denominator = Ve * T(i) - x(i);
if denominator == 0
denominator = 1e-20; % 赋予一个极小的非零值
end
% 预测值计算
x_star = x(i) + Vw * t / sqrt(1 + ((H - y(i)) / denominator)^2);
denominator = Ve * T(i) - x(i);
if denominator == 0
denominator = 1e-10;
end
y_star = y(i) + Vw * t / sqrt(1 + ((Ve * T(i) - x(i)) / denominator)^2);
% 计算中间斜率 M1
M1 = (Ve * T(i) - x(i)) / (H - y(i));
% 计算中间斜率 M2
M2 = (Ve * (T(i)+t) - x_star) / (H - y_star);
% 校正值计算，使用改进的欧拉法公式
x(i + 1) = 0.5 * (x_star + x(i) + Vw * t / sqrt(1 + ((H - y_star) / (Ve * (T(i)+t)
- x_star))^2));
y(i + 1) = 0.5 * (y_star + y(i) + Vw * t / sqrt(1 + ((Ve * (T(i)+t) - x_star) / (H
- y_star))^2));
T(i + 1) = i * t;
% 如果 y 超过 H，则停止循环
if y(i + 1) >= H
k = i; % 更新实际步数
break;
end
end
% 截取实际计算的步数对应的数据
T = T(1:k + 1);
x = x(1:k + 1);
y = y(1:k + 1);
% 创建表格
```

```
resultTable = table(T, x, y, 'VariableNames', {'t', 'x', 'y'});
disp(resultTable);
end
```

Result

```
>> ae2(0.1,10)
    t                  x                    y
    ___    ____________________    ____________________
    0      0                       0
    0.1    2.6807674639146895269   44.839728865955746057
    0.2    10.359329740820363629   82.490963503333572871
    0.3    32.912095866965172246   111.81820696647648988
    0.4    61.873979062800202057   143.76300728866341313
```

得到了更精确的结果，并可以试试用更小步长计算的结果：

Result

```
>> ae2(0.05,10)
    t                  x                    y
    ____    ____________________    ____________________
    0       0                       0
    0.05    0.51867862434450977146  22.488036860797711715
    0.1     2.0158991272675451233   41.644973819026764344
    0.15    4.7043819461667144871   60.72231091574857633
    0.2     8.881713325562459669    79.639978126106153621
    0.25    15.116865250129401544   98.199410513319348068
    0.3     25.710592425763845094   114.97317459482497867
    0.35    28.711480554077469864   134.17624529534353555
```

任务 2

在对由模型直接得到的微分方程组用 Euler 法计算时，我们计算到 $y_k < H$，$y_{k+1} \geq H$ 即停止，然后取 $L \approx x_{k+1}$，但这样的做法可能会有不小误差，即使步长减小而结果仍可能不能改善结果，因此可以采取变步长法：当计算到 $y_k < H < y_{k+1}$，两端距离 H 较远时，以 x_k, y_k 为起点，缩小步长继续计算（例如为原来的十分之一），将改善结果. 试用这种变步长方法来改进在任务 1 中得到的结果.

以下展示的代码可以完成该功能，并可以缩小 5 次步长，以达到更精确的结果：

MATLAB

```
function se1(t, k)
% 设置精度为 10 位
```

```
digits(10);
% 定义常量，使用 vpa 转换为可变精度
H = vpa(120);Ve = vpa(90);Vw = vpa(450);t = vpa(t);
% 初始化数组
x = zeros(k + 1, 1, 'sym');
y = zeros(k + 1, 1, 'sym');
T = zeros(k + 1, 1, 'sym');
% 初始条件
x(1) = vpa(0);y(1) = vpa(0);T(1) = vpa(0);
% 变步长后的迭代计数器
variable_step_iter = 0;
% 用于存储所有结果的表格
all_resultTable = table('Size', [0, 3], 'VariableNames', {'t', 'x', 'y'},
'VariableTypes', {'sym', 'sym', 'sym'});
% 循环计算每一步的 t, x, y
for i = 1:k
% 计算斜率 M
M = (Ve * T(i) - x(i)) / (H - y(i));
% 更新 x、y 和 T 的值
x(i + 1) = x(i) + Vw * t * M / sqrt(1 + M^2);
y(i + 1) = y(i) + Vw * t / sqrt(1 + M^2);
T(i + 1) = T(i) + t;
% 判断是否需要变步长
if y(i) < H && y(i + 1) >= H
x(i+1) = x(i);
y(i+1) = y(i);
T(i+1) = T(i);
t = t/10;
variable_step_iter = variable_step_iter + 1;
if variable_step_iter > 5
break
end
continue
end
% 将当前结果添加到总表格中
new_row = table(T(i + 1), x(i + 1), y(i + 1), 'VariableNames', {'t', 'x', 'y'});
all_resultTable = [all_resultTable; new_row];
end
% 显示最终结果表格
disp(all_resultTable);
end
```

Result

```
>> sel(0.1,100)
```

t	x	y
0.1	0	45.0
0.2	5.361534928	89.67945773
0.21	7.092876771	93.83306525
0.22	8.943688615	97.93483262
0.23	10.93030334	101.9725748
0.24	13.07439894	105.9289436
0.25	15.40629417	109.7776141
0.26	17.97181291	113.4746553
0.27	20.84963789	116.9341529
0.28	24.21353233	119.9231684
0.281	24.66217362	119.9581111
0.282	25.11117534	119.9880687
0.2821	25.15613108	119.990064
0.2822	25.20109017	119.9919823
0.2823	25.24605292	119.9938131
0.2824	25.29101971	119.9955416
0.2825	25.3359911	119.9971459
0.2826	25.38096798	119.9985881
0.2827	25.42595204	119.9997857
0.28271	25.43045169	119.9998423
0.28272	25.43495138	119.9998951
0.28273	25.43945112	119.999943
0.28274	25.44395094	119.999984

Result

```
>> se1(0.05,100)
```

t	x	y
0.05	0	22.5
0.1	1.037357249	44.97607372
0.15	3.412050748	67.35040792
0.2	7.646151587	89.44842571
0.25	14.8678979	110.757963
0.255	16.30058835	112.4928688
0.26	17.79244761	114.1771657
0.265	19.35319608	115.7978335
0.27	20.99713108	117.3340554
0.275	22.74795396	118.7472501
0.28	24.65531996	119.9407554
0.2805	24.87900069	119.9650851
0.281	25.10298382	119.9864522
0.28105	25.12542502	119.9880779

0.2811	25.14786929	119.9896605
0.28115	25.17031682	119.9911962
0.2812	25.19276782	119.9926803
0.28125	25.21522256	119.9941067
0.2813	25.23768139	119.9954673
0.28135	25.26014474	119.996751
0.2814	25.28261326	119.9979408
0.28145	25.30508796	119.9990075
0.2815	25.32757082	119.9998856
0.281505	25.32982055	119.9999202
0.28151	25.33207032	119.9999521
0.281515	25.33432015	119.9999802

如上展示，我们在该算法下获得了更多位数有效数字，从结果看，至少 4 位小数是可靠的。

任务 4

如果当基地发射导弹的同时，敌艇立即由仪器发觉. 假定敌艇为一高速快艇，它即刻以 135km/h 的速度与导弹方向垂直的方向逃逸，问导弹何时何地击中敌艇?试建立数学模型并求解.

正如任务指出，敌艇可以发现并选择自己的逃生测略，以下代码可以计算当敌艇与导弹方向垂直的方向逃逸时的(x,y)坐标随时间变化的情况，并作图可视化了这一过程：

MATLAB

```
% 参数设置
V_m = 450; % 导弹速度
V_s = 135; % 敌艇速度
y_0 = 120; % 敌艇初始 y 坐标
t = 0.0001; % 时间步长
max_steps = 3000; % 最大步数
% 预分配轨迹变量的内存
x_m_trace = zeros(max_steps + 1, 1);y_m_trace = zeros(max_steps + 1, 1);
x_s_trace = zeros(max_steps + 1, 1);y_s_trace = zeros(max_steps + 1, 1);
T_trace = zeros(max_steps + 1, 1);
% 初始化变量
x_m = 0;y_m = 0;x_s = 0;y_s = y_0;T = 0;
index = 1; % 用于记录轨迹数组的索引
for i = 1:max_steps
% 计算导弹指向敌艇的方向向量
dx = x_s - x_m;
dy = y_s - y_m;
dist = sqrt(dx^2 + dy^2);
% 防止除零错误
if dist < eps
```

```
break;
end
% 导弹速度方向的单位向量
unit_dx_m = dx / dist;
unit_dy_m = dy / dist;
% 敌艇速度方向与导弹指向敌艇的方向垂直
unit_dx_s = -unit_dy_m;
unit_dy_s = unit_dx_m;
% 计算敌艇新位置并存储
x_s = x_s + V_s * unit_dx_s * t;
y_s = y_s + V_s * unit_dy_s * t;
x_s_trace(index) = x_s;
y_s_trace(index) = y_s;
% 计算导弹新位置并存储
x_m = x_m + V_m * unit_dx_m * t;
y_m = y_m + V_m * unit_dy_m * t;
x_m_trace(index) = x_m;
y_m_trace(index) = y_m;
% 更新时间和索引
T = T + t;
T_trace(index) = T;
index = index + 1;
% 判断是否击中敌艇，当距离小于某个阈值时认为击中
if sqrt((x_m - x_s)^2 + (y_m - y_s)^2) < 0.05
break;
end
end
% 截取实际使用的轨迹数据
x_m_trace = x_m_trace(1:index - 1);
y_m_trace = y_m_trace(1:index - 1);
x_s_trace = x_s_trace(1:index - 1);
y_s_trace = y_s_trace(1:index - 1);
T_trace = T_trace(1:index - 1);
% 创建表格
all_resultTable = table('Size', [0, 5],'VariableNames', ...
{'时间 t', '导弹 x 坐标', '导弹 y 坐标', '敌艇 x 坐标', '敌艇 y 坐标'}, ...
'VariableTypes', {'sym','sym','sym', 'sym', 'sym'});
for i = 300:300:min(2700, length(T_trace))
new_row = table(T_trace(i), x_m_trace(i),
y_m_trace(i),x_s_trace(i),y_s_trace(i), ...
'VariableNames', {'时间 t', '导弹 x 坐标', '导弹 y 坐标', ...
'敌艇 x 坐标', '敌艇 y 坐标'});
all_resultTable = [all_resultTable; new_row];
end
```

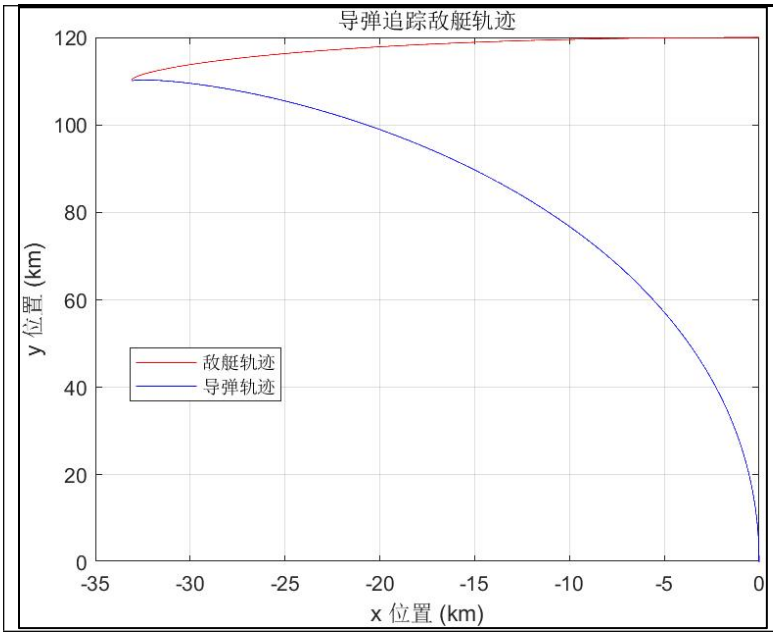
```
% 输出表格
disp(all_resultTable);
% 输出结果
fprintf('导弹击中敌艇的时间: %.4f 小时\n', T);
fprintf('导弹击中敌艇的位置: (%.4f, %.4f) km\n', x_m, y_m);
% 可视化
figure;
plot(x_s_trace, y_s_trace, 'r', 'DisplayName', '敌艇轨迹');
hold on;
plot(x_m_trace, y_m_trace, 'b', 'DisplayName', '导弹轨迹');
xlabel('x 位置 (km)');
ylabel('y 位置 (km)');
title('导弹追踪敌艇轨迹');
legend('Location', 'best');
grid on;
```

Result

```
>> q3_1
    时间 t    导弹 x 坐标    导弹 y 坐标    敌艇 x 坐标    敌艇 y 坐标
    ______    ___________    ___________    ___________    ___________

      0.03       -0.23609         13.497        -4.0492         119.93
      0.06       -0.98652         26.975        -8.0926          119.7
      0.09        -2.3251         40.408        -12.122          119.3
      0.12        -4.3491         53.753        -16.126          118.7
      0.15        -7.1932         66.948        -20.084         117.84
      0.18        -11.056         79.878        -23.964         116.68
      0.21        -16.265         92.323        -27.697         115.12
      0.24        -23.461         103.71        -31.114         112.96

导弹击中敌艇的时间: 0.2666 小时
导弹击中敌艇的位置: (-33.0136, 110.1347) km
```



任务 5

如果敌艇以135km/h 的速度与导弹方向成固定夹角的方向逃逸，问导弹何时何地击中敌艇？试建立数学模型，并选择若干特殊角度进行计算.

写 MATLAB 代码，针对不同的角度进行计算：

MATLAB

```
% 参数设置
V_m = 450; % 导弹速度
V_s = 135; % 敌艇速度
y_0 = 120; % 敌艇初始 y 坐标
t = 0.0001; % 时间步长
max_steps = 5000; % 最大步数
% 选择特殊角度，这里选择 30 度、45 度、60 度
angles = [30, 45, 60];
angles = angles * pi / 180; % 将角度转换为弧度
for angle_idx = 1:length(angles)
theta = angles(angle_idx);
% 预分配轨迹变量的内存
x_m_trace = zeros(max_steps + 1, 1);y_m_trace = zeros(max_steps + 1, 1);
x_s_trace = zeros(max_steps + 1, 1);y_s_trace = zeros(max_steps + 1, 1);
T_trace = zeros(max_steps + 1, 1);
% 初始化变量
x_m = 0;y_m = 0;x_s = 0;y_s = y_0;T = 0;index = 1; % 用于记录轨迹数组的索引
for i = 1:max_steps
% 计算导弹指向敌艇的方向向量
dx = x_s - x_m;
dy = y_s - y_m;
dist = sqrt(dx^2 + dy^2);
```

```
% 防止除零错误
if dist < eps
break;
end
% 导弹速度方向的单位向量
unit_dx_m = dx / dist;
unit_dy_m = dy / dist;
% 敌艇速度方向的单位向量，通过旋转得到
rot_matrix = [cos(theta), -sin(theta); sin(theta), cos(theta)];
unit_d = rot_matrix * [unit_dx_m; unit_dy_m];
unit_dx_s = unit_d(1);
unit_dy_s = unit_d(2);
% 计算敌艇新位置并存储
x_s = x_s + V_s * unit_dx_s * t;
y_s = y_s + V_s * unit_dy_s * t;
x_s_trace(index) = x_s;
y_s_trace(index) = y_s;
% 计算导弹新位置并存储
x_m = x_m + V_m * unit_dx_m * t;
y_m = y_m + V_m * unit_dy_m * t;
x_m_trace(index) = x_m;
y_m_trace(index) = y_m;
% 更新时间和索引
T = T + t;
T_trace(index) = T;
index = index + 1;
% 判断是否击中敌艇，当距离小于某个阈值时认为击中
if sqrt((x_m - x_s)^2 + (y_m - y_s)^2) < 0.05
break;
end
end
% 截取实际使用的轨迹数据
x_m_trace = x_m_trace(1:index - 1);
y_m_trace = y_m_trace(1:index - 1);
x_s_trace = x_s_trace(1:index - 1);
y_s_trace = y_s_trace(1:index - 1);
T_trace = T_trace(1:index - 1);
% 创建表格，将数据类型设置为双精度浮点数
all_resultTable = table('Size', [0, 5], 'VariableNames', ...
{'时间 t', '导弹 x 坐标', '导弹 y 坐标', '敌艇 x 坐标', '敌艇 y 坐标'}, ...
'VariableTypes', {'double', 'double', 'double', 'double', 'double'});
for i = 300:300:min(3000, length(T_trace))
new_row = table(T_trace(i), x_m_trace(i), y_m_trace(i), x_s_trace(i),
y_s_trace(i), ...
```

```
'VariableNames', {'时间 t', '导弹 x 坐标', '导弹 y 坐标', ...
'敌艇 x 坐标', '敌艇 y 坐标'});
all_resultTable = [all_resultTable; new_row];
end
% 设置小数输出格式
format short
% 输出结果
fprintf('夹角为 %.0f 度时：\n', theta * 180 / pi);
fprintf('导弹击中敌艇的时间: %.4f 小时\n', T);
fprintf('导弹击中敌艇的位置: (%.4f, %.4f) km\n', x_m, y_m);
% 输出表格
disp(all_resultTable);
% 可视化
figure;
plot(x_s_trace, y_s_trace, 'r', 'DisplayName', '敌艇轨迹');
hold on;
plot(x_m_trace, y_m_trace, 'b', 'DisplayName', '导弹轨迹');
xlabel('x 位置 (km)');
ylabel('y 位置 (km)');
title(['导弹追踪敌艇轨迹，夹角为 ', num2str(theta * 180 / pi), ' 度']);
legend('Location', 'best');
grid on;
end
```

Result

```
>> q5
夹角为 30 度时：
导弹击中敌艇的时间: 0.3602 小时
导弹击中敌艇的位置: (-31.5241, 155.7419) km
    时间 t     导弹 x 坐标     导弹 y 坐标     敌艇 x 坐标     敌艇 y 坐标

    ____    ___________    ___________    ___________    ___________

    0.03       -0.11682         13.499        -2.0552         123.49
    0.06       -0.48245         26.994        -4.1745         126.94
    0.09         -1.1217         40.479        -6.3632         130.35
    0.12         -2.0646         53.945        -8.6282         133.71
    0.15         -3.3489         67.384         -10.978            137
    0.18         -5.0224         80.779         -13.422         140.23
    0.21         -7.1487          94.11         -15.974         143.38
    0.24          -9.816         107.34         -18.652         146.42
    0.27         -13.155         120.42         -21.481         149.31
     0.3         -17.383         133.24         -24.502         152.01

夹角为 45 度时：
导弹击中敌艇的时间: 0.3384 小时
```

导弹击中敌艇的位置：(-38.2163, 142.0517) km

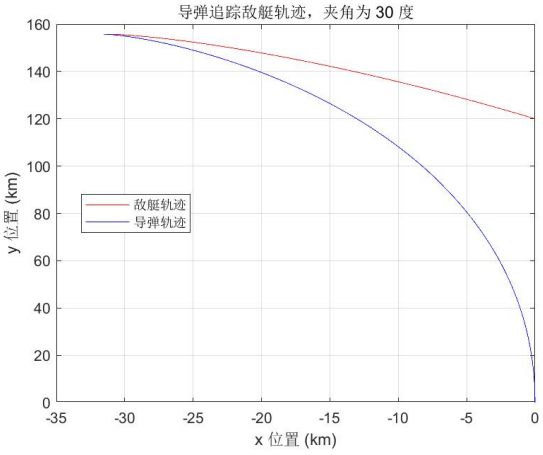
时间 t	导弹 x 坐标	导弹 y 坐标	敌艇 x 坐标	敌艇 y 坐标
0.03	-0.16552	13.499	-2.8986	122.83
0.06	-0.68498	26.988	-5.8704	125.58
0.09	-1.5962	40.457	-8.9208	128.24
0.12	-2.9458	53.889	-12.056	130.81
0.15	-4.7925	67.261	-15.285	133.25
0.18	-7.2128	80.541	-18.615	135.56
0.21	-10.31	93.679	-22.059	137.69
0.24	-14.235	106.59	-25.631	139.59
0.27	-19.22	119.13	-29.349	141.19
0.3	-25.69	130.97	-33.232	142.33

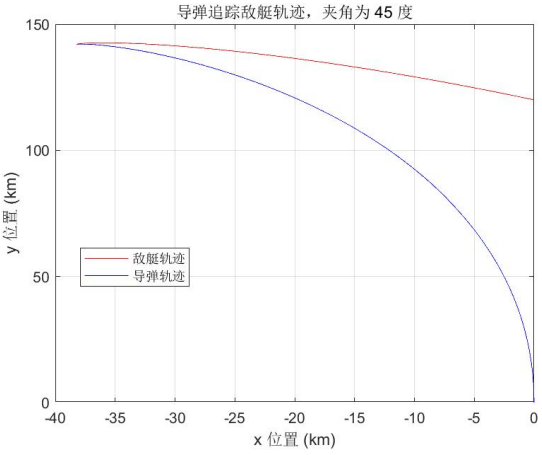
夹角为 60 度时：

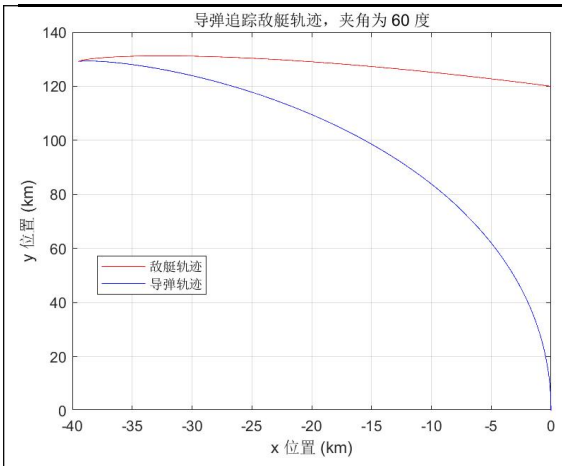
导弹击中敌艇的时间：0.3137 小时

导弹击中敌艇的位置：(-39.4655, 129.1439) km

时间 t	导弹 x 坐标	导弹 y 坐标	敌艇 x 坐标	敌艇 y 坐标
0.03	-0.20323	13.498	-3.5374	121.97
0.06	-0.84332	26.982	-7.1367	123.83
0.09	-1.9713	40.434	-10.801	125.55
0.12	-3.6512	53.828	-14.533	127.13
0.15	-5.9654	67.127	-18.335	128.52
0.18	-9.0244	80.273	-22.209	129.7
0.21	-12.984	93.176	-26.155	130.6
0.24	-18.083	105.67	-30.166	131.15
0.27	-24.73	117.4	-34.212	131.19
0.3	-33.846	127.27	-38.143	130.3







任务 6

数据分析

敌艇逃逸方向与导弹方向夹角	导弹击中敌艇时间（小时）
30°	0.3602
45°	0.3384
60°	0.3137
90°	0.2666

分析：

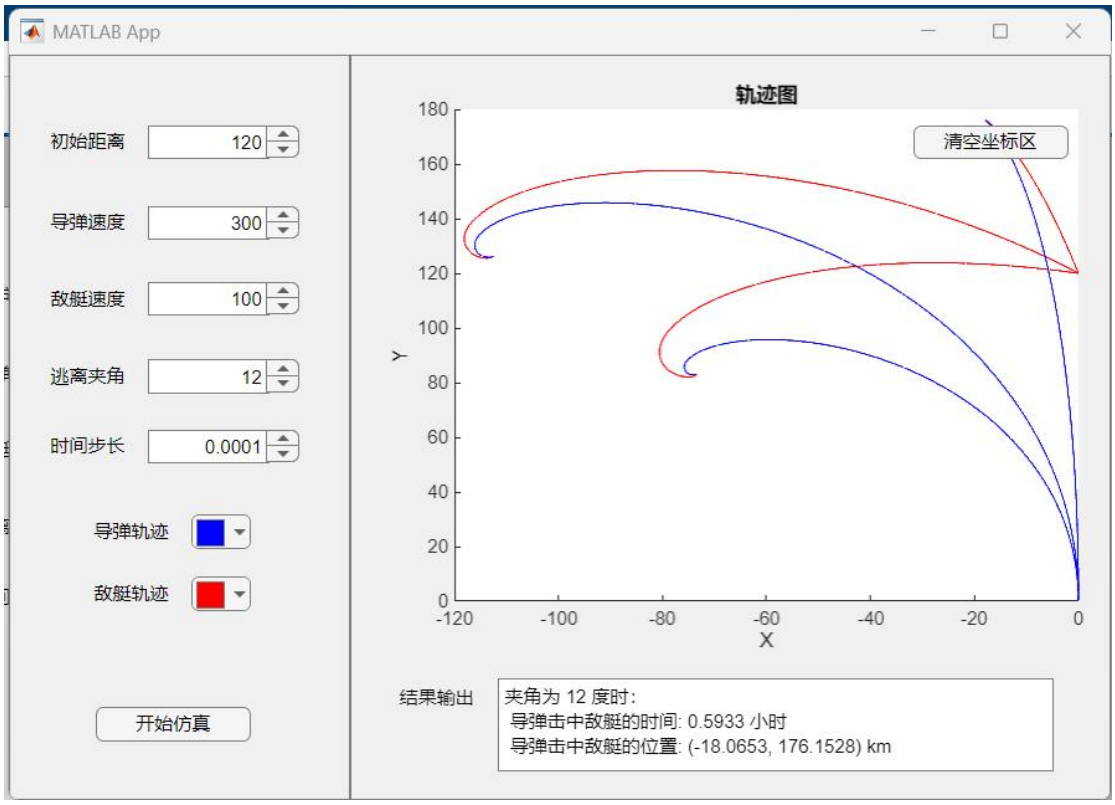
逃逸夹角与被击中时间的关系：从表格数据可以明显看出，随着敌艇逃逸方向与导弹方向夹角**的增大，导弹击中敌艇所需的时间逐渐减少 。当夹角为 90 度时，敌艇被击中的时间最短。这是因为当夹角越大，敌艇在垂直于导弹初始追踪方向上的速度分量相对越大，能更快地拉开与导弹的距离，**从而使得导弹需要更长的路径才能击中敌艇，导致击中时间缩短。

敌艇逃逸策略：对于敌艇而言，若要尽可能延长躲避时间，应选择与导弹方向夹角较小的方向逃逸，即尽量朝着远离导弹初始指向的方向 “斜着跑”，而非垂直于导弹方向逃逸。但实际情况中，敌艇的逃逸方向选择还会受到地形、自身性能限制等多种因素影响 。

导弹追踪效率：对于导弹追踪系统来说，敌艇逃逸方向夹角的变化会影响击中时间，说明在设计追踪算法和系统时，需要充分考虑目标可能的逃逸方向，以便更精准地预测和追踪目标，提高击中效率 。

*特殊任务：

制作了 mlapp 模拟追及过程，放在本文档里。



使用说明

本 MATLAB App 旨在模拟导弹追击敌艇的过程，通过设置相关参数，直观展示导弹与敌艇的运动轨迹，并输出导弹击中敌艇的时间和位置信息，帮助用户理解追击过程中的动态变化和相关原理。

界面介绍

（一）参数设置区

初始距离：用于设定导弹与敌艇初始时刻在某一方向上的距离，单位为千米（km），默认值为 120 km 。用户可通过输入框直接输入数值或使用微调按钮调整。

导弹速度：设置导弹的飞行速度，单位为千米每小时（km/h），默认值为 135 km/h 。同样可通过输入框和微调按钮进行修改。

敌艇速度：指定敌艇的行驶速度，单位为千米每小时（km/h），默认值为 90km/h ，调整方式同前。

逃离夹角：表示敌艇逃离方向与初始相对方向（如导弹指向敌艇方向）的夹角，单位为度（°），默认值为 45° ，可进行输入和微调。

时间步长：定义模拟过程中时间推进的间隔，单位为小时（h），默认值为 0.0001 h ，用户可按需调整。

（二）轨迹显示区

以二维坐标图形式展示导弹和敌艇的运动轨迹。横坐标（X）和纵坐标（Y）表示位置，单位为千米（km）。图中蓝色线条代表导弹轨迹，红色线条代表敌艇轨迹。

清空坐标区” 按钮：点击该按钮可清除当前显示的轨迹图，便于进行新的模拟。

（三）颜色设置区

导弹轨迹：提供下拉菜单，用户可选择不同颜色来设置导弹轨迹的显示颜色，默认为蓝色。

敌艇轨迹：同样是下拉菜单，用于选择敌艇轨迹的显示颜色，默认为红色。

（四）操作控制区

开始仿真按钮：点击此按钮，App 将根据用户设置的参数开始模拟导弹追击敌艇的过程，并在轨迹显示区绘制轨迹，在结果输出区显示相关结果。

（五）结果输出区

显示模拟的结果信息，包括当前逃离夹角数值，以及导弹击中敌艇的时间（单位：小时）和位置（单位：千米，以坐标形式展示，如（x, y））。

三、使用步骤

参数设置：根据实际需求，在参数设置区调整初始距离、导弹速度、敌艇速度、逃离夹角和时间步长等参数。

轨迹颜色选择（可选）：若需要改变导弹和敌艇轨迹的显示颜色，在颜色设置区通过下拉菜单进行选择。

开始模拟：点击 “开始仿真” 按钮，App 将进行模拟计算，并在轨迹显示区绘制导弹和敌艇的运动轨迹，同时在结果输出区显示导弹击中敌艇的时间和位置信息。

清空坐标区：若要进行新的模拟，可点击 “清空坐标区” 按钮清除之前的轨迹图，然后重复上述步骤设置新参数并开始模拟。

结语

本实验通过对导弹跟踪敌艇问题的深入研究，利用数学建模和数值计算方法，成功求解了多种情况下导弹击中敌艇的时间和位置。从基本追踪问题出发，运用 Euler 法及其改进方法，以及变步长法进行计算，不断提高计算精度。同时，针对敌艇不同的逃逸策略，包括垂直逃逸和与导弹方向成固定夹角逃逸，分别建立模型并求解，深入分析了逃逸夹角与击中时间的关系。

研究结果表明，敌艇逃逸方向与导弹方向夹角对击中时间有显著影响，夹角越大，击中时间越短。这为敌艇制定逃逸策略和导弹追踪系统的设计提供了重要参考。然而，实际的军事场景更为复杂，敌艇的逃逸决策还会受到地形、自身性能等多种因素的制约，导弹追踪系统也需要综合考虑更多的实际因素以提高追踪效率。

未来研究可以进一步拓展，例如考虑导弹和敌艇的机动性能变化、环境干扰因素对追踪的影响等，使模型更加贴近实际情况。同时，随着计算技术的不断发展，探索更高效的数值计算方法和优化算法，以提升导弹追踪问题的求解精度和速度，为军事领域的决策和技术发展提供更有力的支持。