# 数学实验作业三

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 刘兆宇 | **班级** | 大数据管理与应用4班 |
| **学号** | 202330860601 | **时间** | 2024.04.11 |

1. **实验目的和内容**
2. 使用MATLAB内置的ode45求解器，求解给定非线性常微分方程组的数值解，理解和掌握常微分方程数值解法的基本原理与方法，同时探索两组不同的初始条件下解的变化情况。
3. 通过缉私舰追赶走私船的动态追击情境，在实际问题中深入理解微分方程在描述和分析动态系统行为中的作用。通过建立适当的微分方程模型，掌握利用数值方法解决实际物理问题的基本技巧。
4. **实验过程**
5. **用ode函数求解给定的微分方程，讨论解的变化情况**
   * + 1. 问题分析：

对于上述非线性微分方程组，应用MATLAB的ode45函数。ode45是一个基于四阶和五阶Runge-Kutta公式的常微分方程求解器，适用于求解非刚性问题，较为通用，使用它来处理上述微分方程组，并观察在不同初始条件下解随tspan的变化情况。

1. 定义微分方程函数

function dxdt = myODEs(t, x)

dxdt = zeros(3,1);

dxdt(1) = x(1) \* (1 - x(1) - x(2) - 6\*x(3));

dxdt(2) = x(2) \* (1.5\*x(1) - x(2) - x(3));

dxdt(3) = x(3) \* (-1 + 3\*x(1) + 0.5\*x(2));

end

1. 求解第一组初始条件：
2. 求解第二组初始条件：
3. plot可视化结果
   * + 1. 伪代码：

定义 myODEs(t, x):

dxdt[1] = x[1] \* (1 - x[1] - x[2] - 6x[3])

dxdt[2] = x[2] \* (1.5x[1] - x[2] - x[3])

dxdt[3] = x[3] \* (-1 + 3x[1] + 0.5x[2])

返回 dxdt

初始条件1: t = [0, 30], x0 = [0.12, 0.003, 0.01]

(t1, x1) = ode45(myODEs, t, x0)

初始条件2: t = [0, 133], x0 = [0.01, 0.00001, 0.001]

(t2, x2) = ode45(myODEs, t, x0)

plot:

绘制(t1, x1) 并标记

绘制(t2, x2) 并标记

设置图例、标题、坐标轴标签

显示图形

* + - 1. 代码：

t = [0 30];

x0 = [0.12; 0.003; 0.01];

[t1, x1] = ode45(@myODEs, t, x0);

t = [0 133];

x0 = [0.01; 0.00001; 0.001];

[t2, x2] = ode45(@myODEs, t, x0);

% 绘制第一组初始条件的解

figure;

plot(t1, x1(:,1), 'r', 'LineWidth', 2);

hold on;

plot(t1, x1(:,2), 'g', 'LineWidth', 2);

plot(t1, x1(:,3), 'b', 'LineWidth', 2);

hold off;

title('\bf第一组初始条件的解');

legend({'$x(t)$', '$y(t)$', '$z(t)$'}, 'Interpreter', 'latex');

xlabel('$t$', 'Interpreter', 'latex')

ylabel('$x, y, z$', 'Interpreter', 'latex');

grid on; grid minor;

% 绘制第二组初始条件的解

figure;

plot(t2, x2(:,1), 'r', 'LineWidth', 2);

hold on;

plot(t2, x2(:,2), 'g', 'LineWidth', 2);

plot(t2, x2(:,3), 'b', 'LineWidth', 2);

hold off;

title('\bf第二组初始条件的解');

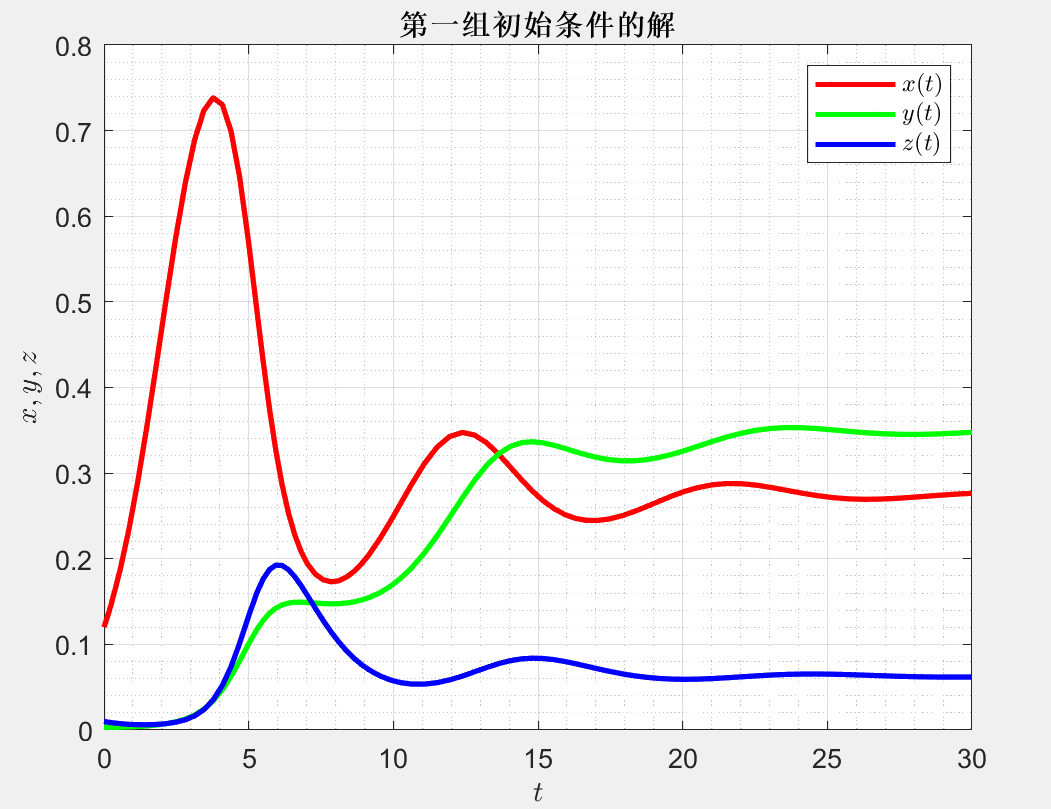
legend({'$x(t)$', '$y(t)$', '$z(t)$'}, 'Interpreter', 'latex');

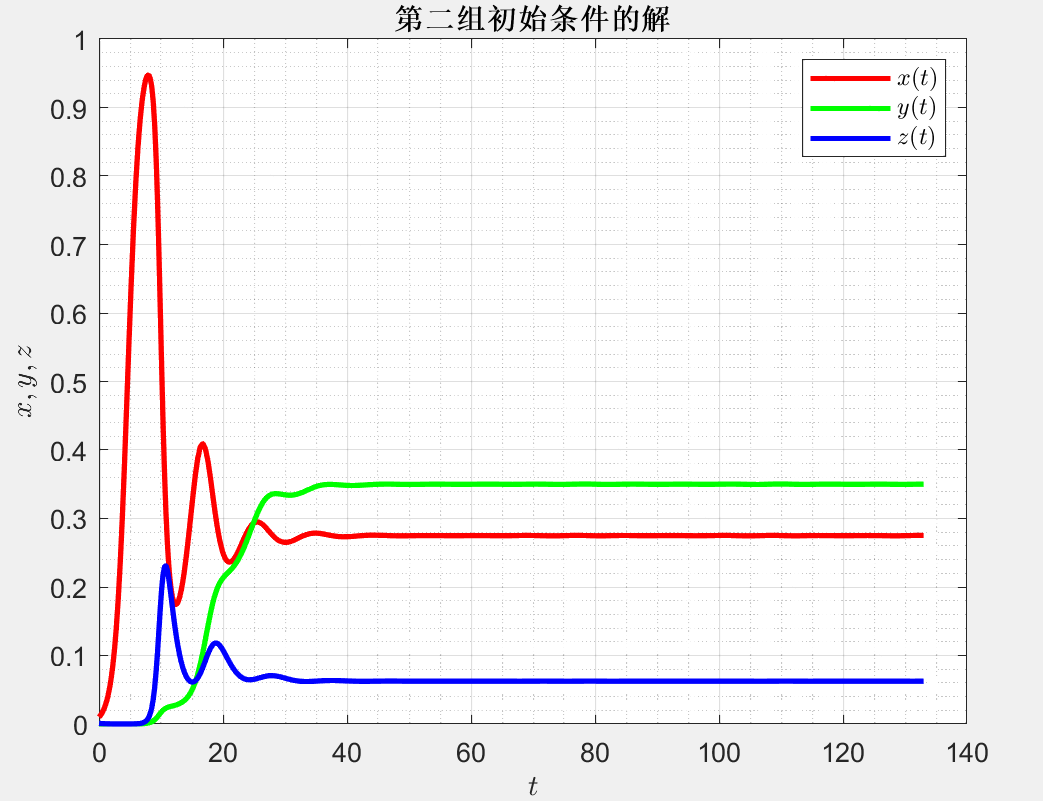
xlabel('$t$', 'Interpreter', 'latex')

ylabel('$x, y, z$', 'Interpreter', 'latex');

grid on; grid minor;

* + - 1. 运行结果





* + - 1. 结果分析：
* **第一组初值及求解区间**：

1. 开始时迅速上升，达到一个峰值后迅速下降，之后逐渐趋于平稳。这个峰值可能表示系统在一个特定的变量或状态下的短暂不稳定性。
2. 开始时略有上升，随后经历一段平稳后进一步上升并逐步趋向于一个较为平稳的状态，但仍有轻微的波动。
3. 开始时略有上升，随后下降并逐步在较低的值处波动并趋于稳定.。

* 这张图显示，解在短期内经历了快速的变化后趋于稳定。每个状态变量似乎都趋向于某个固定值或轻微波动的平衡状态。
* **第二组初值及求解区间**：

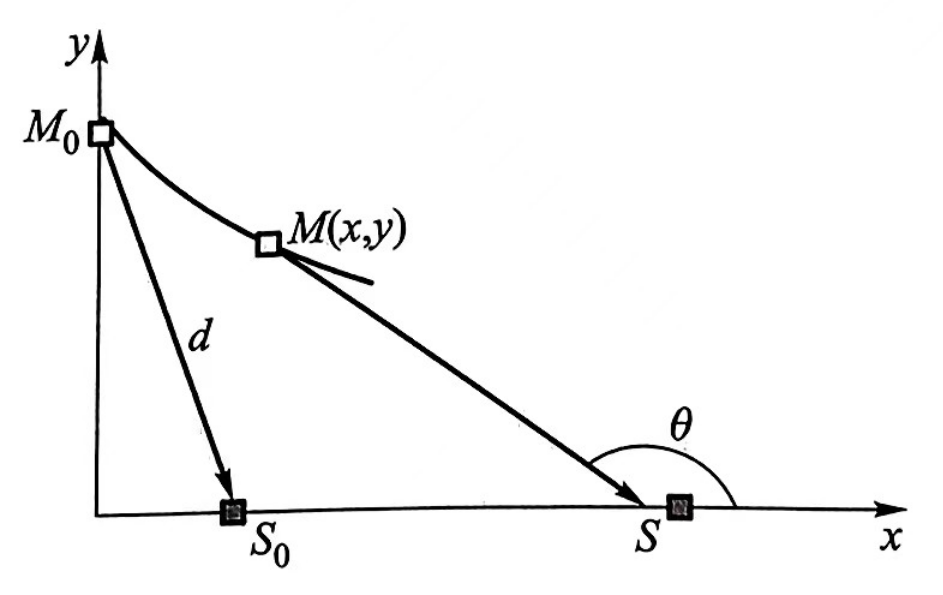
1. 类似于第一张图，开始时迅速上升并达到一个高峰，随后下降。但与第一组条件不同，在经过数次显著的波动后，趋于一个较为平稳的状态。
2. 在开始时逐步波动上升，最后上升到一个相对平稳的值并保持较为平稳的状态。
3. 整体与的波动趋势类似，先迅速上升再下降，经过几次波动后下降并逐步在较低的值趋于稳定.。

* 第二张图揭示了解的系统可能存在一个更长的调整期，最终似乎也趋于某种平衡状态，但相比第一组条件，这一组条件下的波动更加显著。

1. **微分方程建模——追击曲线**
   * + 1. 问题分析

这是一个经典的追及问题；一开始缉私舰与走私船相距d，走私船以匀速a沿直线行驶，缉私舰以匀速v追赶，但保持船的瞬时速度方向始终指向走私船。

1. 数学描述：在本实验中，设走私船沿x轴正方向以速度a移动，其初始位置位于原点(0, 0)。缉私舰从点(0, d)出发，即y轴上距离原d的位置，以最大速*v*向走私船追去。



1. 微分方程建立：

· 假设走私船在任意时刻的位置为，缉私舰的位置为；

· 缉私舰的速度矢量始终指向走私船，因此其速度矢量的方向与从缉私舰到走私船的位矢相同；缉私舰的速度分量，其中*θ*是速度矢量与水平方向的夹角；

· 利用相似三角形的原理，速度分量和位矢分量的比例相同：

· 由速度和位移的导数关系，将上式转化为微分方程：

· 给出初值条件：

1. 设置参数a,v,d，用ode45求数值解；
2. 参数影响分析：实验将通过改变走私船速度*a*、缉私舰速度*v*以及初始距离*d*来分析这些参数如何影响追击轨迹，参数的变化将展示不同条件下缉私舰追赶走私船的效率和轨迹形态。
   * + 1. 代码：

function chasing\_boat

% 设定参数

a = 5; % 走私船沿x轴的速度

v = 15; % 缉私舰的速度

d = 50; % 缉私舰与走私船在y轴上的初始距离

t = [0, 150];

y0 = [0; d];

% 设置ode选项

options = odeset('Events', @(t, y) eventFunction(t, y, a));

% 求解微分方程

[T, Y, TE, YE, IE] = ode45(@(t, y) boatDynamics(t, y, a, v), t, y0, options);

% 绘制轨迹

figure;

plot(Y(:,1), Y(:,2), 'b-', 'LineWidth', 2); hold on;

plot(a\*T, zeros(size(T)), 'r--', 'LineWidth', 2);

xlabel('$x$', 'Interpreter', 'latex');

ylabel('$y$', 'Interpreter', 'latex');

title('缉私舰和走私船的追击轨迹');

axis equal;

grid on; grid minor;

% 标注a, v, d的值

text(20, 13, ['$a=$ ' num2str(a)], 'Interpreter','latex', 'FontSize', 12);

text(20, 9, ['$v=$ ' num2str(v)], 'Interpreter','latex', 'FontSize', 12);

text(20, 5, ['$d=$ ' num2str(d)], 'Interpreter','latex', 'FontSize', 12);

% 终止事件，标出缉私舰追上走私船的位置

if ~isempty(TE)

endMarker = plot(YE(:,1), YE(:,2), 'ko', 'MarkerSize',7, 'LineWidth',2);

legend(endMarker, {'起点和终点'});

fprintf('追击时间： t = %.2f seconds.\n', TE);

end

legend('缉私舰轨迹', '走私船轨迹', '起点和终点', 'Location', 'best');

end

function dydt = boatDynamics(t, y, a, v)

x = y(1);

y = y(2);

xt = a\*t;

yt = 0;

dis = sqrt((xt - x)^2 + y^2);

dxdt = v \* (xt - x) / dis;

dydt = v \* (-y) / dis;

dydt = [dxdt; dydt];

end

function [value,isterminal,direction] = eventFunction(t, y, a)

value = y(1) - a\*t;

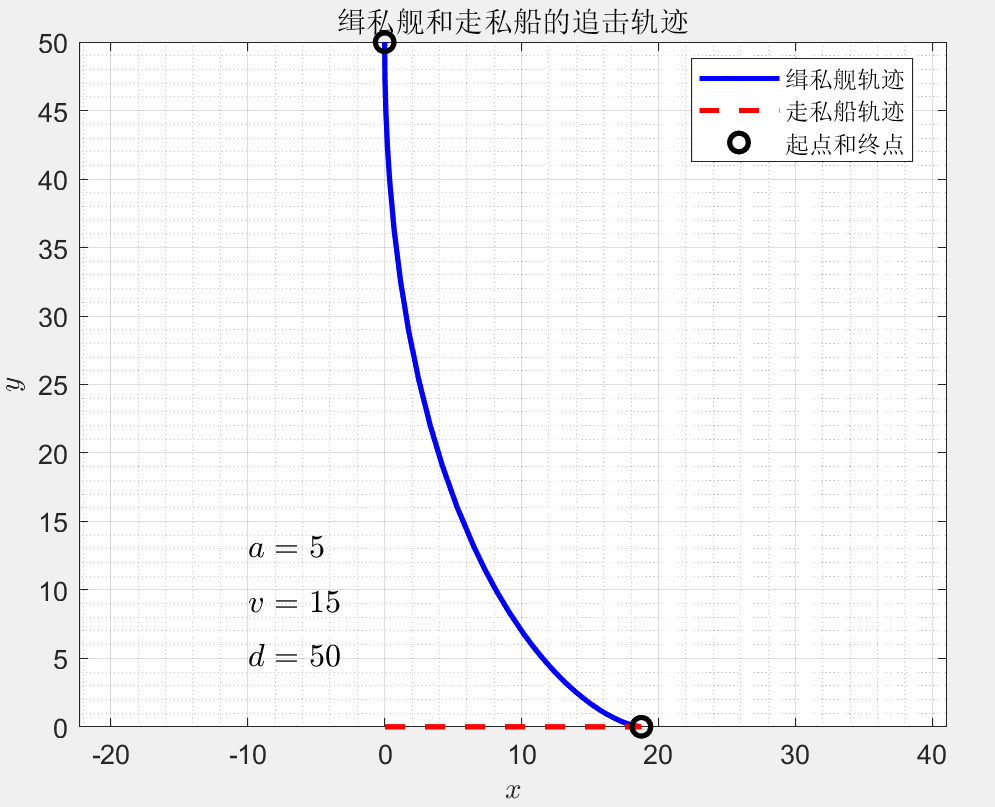
isterminal = 1;

direction = 0;

end

* + - 1. 程序运行与结果分析：

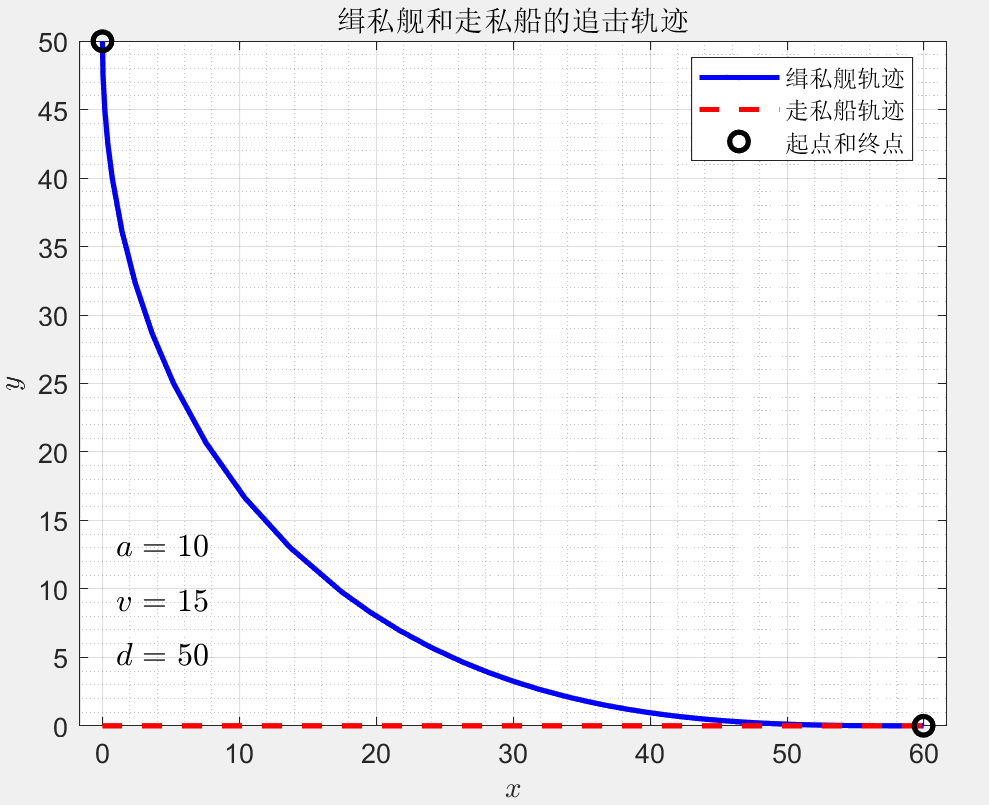




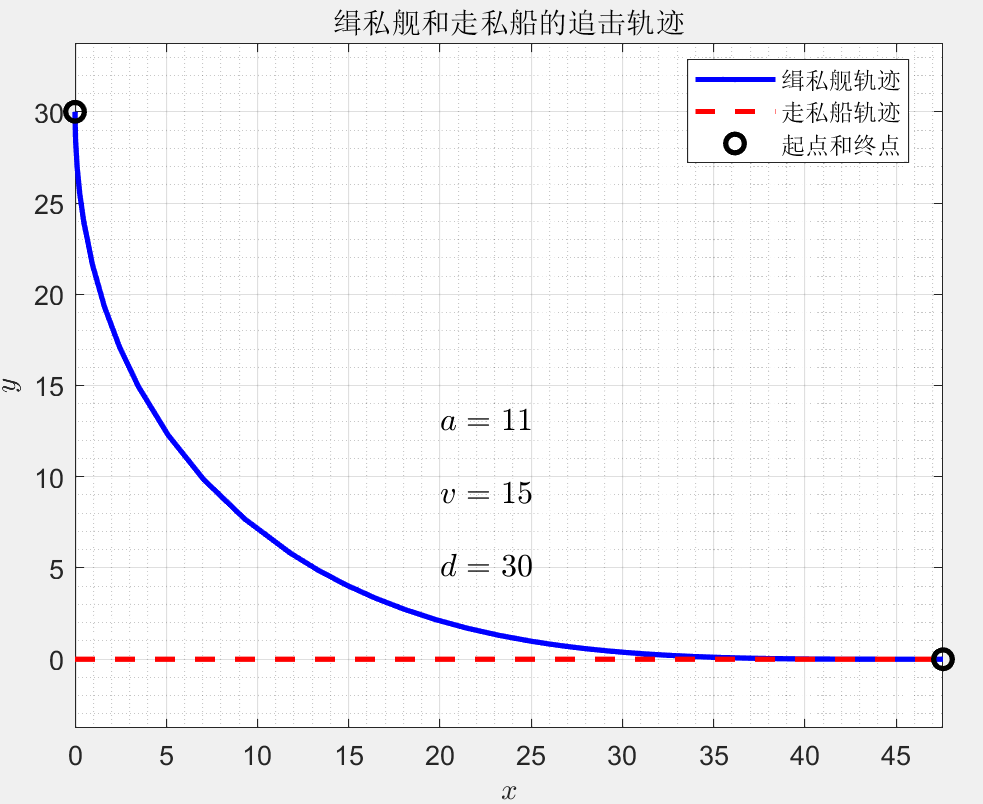


当走私船的速度相对较低而缉私舰的速度相对较高时，缉私舰能够迅速且直接地追上走私船，追及路径较直且追击时间短。这种情况下，速度优势明显，使得缉私舰不需要太长时间便能成功拦截走私船。





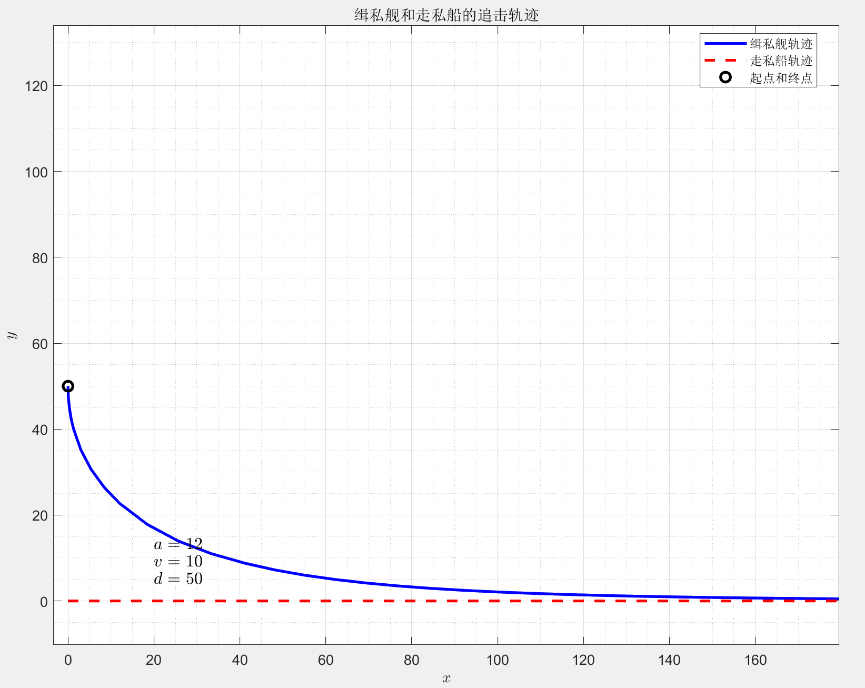






当走私船的速度接近或略低于缉私舰的速度时，缉私舰需要更长的时间来减少与走私船之间的距离，追击轨迹会出现明显的弯曲，尤其是在追击的初期阶段。





如果*a*接近或超过*v*，缉私舰可能永远无法追上走私船。缉私舰的速度*v*是追击成功的关键。只有当*v*明显大于*a*时，缉私舰才有可能在较短时间内追上走私船。初始距离*d*决定了缉私舰开始追击时与走私船的距离。距离越远，即使缉私舰速度高于走私船，也需要更长时间才能追上。