```
%用微分方程解龙头位置

figure;
hold on;
axis equal; % 保持坐标比例一致

% 定义常数
b = 0.55 * 16;        % 起始半径
a = 0.55 / (2 * pi); % 每圈增加的半径

% 微分方程定义
odefun = @(t, y) 1 / sqrt((b - a * y)^2 + a^2);
tspan = [0 300];      % 时间跨度
y0 = 0;               % 初始条件

% 使用ode45求解微分方程
options = odeset('RelTol',1e-5,'AbsTol',1e-7);
[t, y] = ode45(odefun, tspan, y0, options);

% 创建每秒一个时间点的向量
t_interp = 0:1:300;  % 每秒一个点

% 插值得到每秒的解
y_interp = -interp1(t, y, t_interp, 'linear');

% 计算等距螺旋线的坐标
theta_spiral = linspace(0, -32*pi, 8000);
r_spiral = b + a * theta_spiral;
x_spiral = r_spiral .* cos(theta_spiral);
y_spiral = r_spiral .* sin(theta_spiral);

% 将插值解映射到螺旋线的坐标上
theta_solution = y_interp;  % 使用插值解作为 theta
r_solution = b + a * theta_solution;  % 根据解的角度计算半径
x_solution = r_solution .* cos(theta_solution);  % x坐标
y_solution = r_solution .* sin(theta_solution);  % y坐标

% 绘制等距螺旋线
plot(x_spiral, y_spiral, 'b')

% 绘制插值解的点
plot(x_solution, y_solution, 'ro')

% 现有螺旋线的参数
b = 0.55 * 16;        % 起始半径
a = 0.55 / (2 * pi); % 每圈增加的半径

% 现有螺旋线的计算和绘制
theta_spiral = linspace(0, -32*pi, 4000);
r_spiral = b + a * theta_spiral;
x_spiral = r_spiral .* cos(theta_spiral);
y_spiral = r_spiral .* sin(theta_spiral);
plot(x_spiral, y_spiral, 'b-'); % 绘制蓝色实线

% 更大螺旋线的计算和绘制
b_larger = b + 8.8;  % 增大起始半径
r_spiral_larger = b_larger + a * (theta_spiral - 20 * pi);
x_spiral_larger = r_spiral_larger .* cos(theta_spiral);
```

```
y_spiral_larger = r_spiral_larger .* sin(theta_spiral);
plot(x_spiral_larger, y_spiral_larger, 'k--'); % 绘制黑色虚线

% 绘制原始的散点（圆心）
plot(x_solution, y_solution, 'ro'); % 使用红色圆圈标记


hold off; % 解除保持状态
xlabel('X');
ylabel('Y');
grid on; % 添加网格以便更清楚地查看
```

```
%                          问题一：模拟舞龙队盘入的过程                                  %
%  （运行前请clear工作区,避免出现错误！）

% 参数设置
benches_num = 224;  % 板凳数量
head_long = 3.41; % 龙头长度
body_long = 2.20; % 龙身和龙尾长度
benches_width = 0.3; % 板宽
hole_to_head = 0.275; % 板凳孔距离最近板凳头距离
p = 0.55;           % 螺距
v_head = 1.0;       % 龙头速度
T = 300;            % 模拟时间
dt = 0.01;          % 时间步长

% 第一个把手的初始位置和角度设置
theta0 = 2*pi*16;
r0 = p*16;
% 初始化龙头的位置
positions(1, :, 1) = [r0 * cos(theta0), r0 * sin(theta0)];
% 计算每个板凳孔相对于上一节的偏移
L = [head_long - 2 * hole_to_head; ...
    repmat(body_long - 2 * hole_to_head, benches_num-1, 1)];  % 每节的长度

% 初始化龙身和龙尾位置
initial_theta = theta0;
initial_r = r0;

%计算龙身的初始位置
for i = 2:benches_num
    delta_theta(i) = L(i-1) / initial_r;  % 每节之间的角度差 弧长=半径×角度
    initial_theta = initial_theta + delta_theta(i);
    initial_r = p / (2 * pi) * initial_theta;     % 半径变化
    positions(i, 1, 1) = initial_r * cos(initial_theta);  % x位置
    positions(i, 2, 1) = initial_r * sin(initial_theta);  % y位置
end

%初始化速度矩阵
velocities = zeros(224, 301);

%初始化角度和半径
current_theta = theta0;
current_r = r0;

%每一个dt时刻的龙位置
for j = dt:dt:T
    t = round(j / dt);
    % 计算龙头位置
    theta_head = current_theta - v_head * dt / current_r;
    r_head = p / (2 * pi) * theta_head;
    positions(1, :, t+1) = [r_head * cos(theta_head), ...
                            r_head * sin(theta_head)];
    % 更新龙头前把手极坐标的角度和半径
    current_theta = theta_head;
    current_r = r_head;
    % 更新第一节龙身前把手的角度和半径
    initial_theta = current_theta;
    initial_r = current_r;
```

```
    % 计算当前时刻龙身和龙尾位置
    for i = 2:benches_num
        delta_theta(i) = L(i-1) / initial_r;  % 近似：角度=弧长/半径
        initial_theta = initial_theta + delta_theta(i);
        initial_r = p / (2 * pi) * initial_theta;
        positions(i, 1, t+1) = initial_r * cos(initial_theta);  % x位置
        positions(i, 2, t+1) = initial_r * sin(initial_theta);  % y位置
    end

    % 计算每节的速度
    if t > 0
        a = (positions(:, 1, t+1) - positions(:, 1, t)) / dt;  % vx
        b = (positions(:, 2, t+1) - positions(:, 2, t)) / dt;  % vy
        velocities(:, t+1) = sqrt(a.^2 + b.^2);
    end


    % %图像显示占用电脑资源影响运行速度，可以选择注释提高运行效率
    % % 绘制当前时刻龙的位置
    % pause(0.01);
    % clf;
    % hold on;
    % axis equal;
    % xlabel('X (米)');
    % ylabel('Y (米)');
    % % 设置坐标轴范围
    % xlim([-12, 12]);
    % ylim([-12, 12]);
    % title(['板凳龙行进示意图 (t = ', num2str(j), 's)']);
    % grid on;
    % % 画背景螺线图
    % theta_spiral = linspace(0, -32*pi, 10000);
    % r_spiral = 0.55 * 16 + (0.55 / (2 * pi)) * theta_spiral;
    % x_spiral = r_spiral .* cos(theta_spiral);
    % y_spiral = r_spiral .* sin(theta_spiral);
    % plot(x_spiral, y_spiral,'LineWidth', 0.5, 'Color', 'm');
    % %画龙
    % plot(positions(1, 1, t+1), positions(1, 2, t+1), 'ro-', ...
    %     'MarkerSize', 4, 'LineWidth', 2,'MarkerFaceColor', 'r');
    % plot(positions(2:end, 1, t+1), positions(2:end, 2, t+1),...
    %     'co-', 'MarkerSize', 4, 'LineWidth', 2,'MarkerFaceColor', 'b');
    % line([positions(1, 1, t+1), positions(2, 1, t+1)],...
    %     [positions(1, 2, t+1), positions(2, 2, t+1)],...
    %     'Color','red','LineWidth', 2,'LineStyle','-');
    % hold off;


end

% 输出0s - 300s数据
output_times = 0:1:300;
positions_output = zeros(benches_num, 2, length(output_times));
velocities_output = zeros(benches_num,length(output_times));

%存入数据
for i = 1:length(output_times)
    t_idx = round(output_times(i)/dt) + 1;
    positions_output(:, :, i) = positions(:, :, t_idx);
```

```
    velocities_output(:, i) = velocities(:, t_idx);
end
velocities_output(:, 1)=1;

% 保存结果到Excel文件
filename = 'result1.xlsx';
location=[];
for i = 1:length(output_times)
    location = [location,reshape(squeeze(positions_output(:, :, i))'...
                ,1,448)'];
end

writematrix(location, filename, 'Sheet','位置', 'Range', 'B2');
writematrix(velocities_output, filename, 'Sheet','速度', 'Range', 'B2');
disp('数据已存入result1.xlsx');
```

```
% 函数功能：检测两条线段是否相交，后续问题需要判断是否相交

function is_intersecting = check_intersection(P1, P2, Q1, Q2)

    cross_product = @(A, B) A(1)*B(2) - A(2)*B(1);

    P1P2 = P2 - P1;
    P1Q1 = Q1 - P1;
    P1Q2 = Q2 - P1;

    Q1Q2 = Q2 - Q1;
    Q1P1 = P1 - Q1;
    Q1P2 = P2 - Q1;

    d1 = cross_product(P1P2, P1Q1);
    d2 = cross_product(P1P2, P1Q2);
    d3 = cross_product(Q1Q2, Q1P1);
    d4 = cross_product(Q1Q2, Q1P2);

    % 如果符号不同，表示相交
    is_intersecting = (d1 * d2 < 0) && (d3 * d4 < 0);
end
```

```
%                                问题二：板凳龙碰撞                                %
%  (运行前请clear工作区,避免出现错误!)

% 参数设置
benches_num = 224;
head_long = 3.41;
body_long = 2.20;
benches_width = 0.3;
hole_to_head = 0.275;
p = 0.55;
v_head = 1.0;


%由问题一知前300s不会碰撞，假设再过200秒之内会碰撞
T = 500;    % 模拟时间
%缩小时间步长可以提高精度
dt = 0.01; % 时间步长

%需要实现龙在螺旋线上的动态显示，与第一问类似，不再注释
theta0 = 2*pi*16;
r0 = p*16;
positions(1, :, 1) = [r0 * cos(theta0), r0 * sin(theta0) ];
L = [head_long - 2 * hole_to_head; ...
    repmat(body_long - 2 * hole_to_head, benches_num-1, 1)];
initial_theta = theta0;
initial_r = r0;

for i = 2:benches_num
    delta_theta(i) = L(i-1) / initial_r;
    initial_theta = initial_theta + delta_theta(i);
    initial_r = p / (2 * pi) * initial_theta;
    positions(i, 1, 1) = initial_r * cos(initial_theta);
    positions(i, 2, 1) = initial_r * sin(initial_theta);
end

velocities = zeros(224, 301);
current_theta = theta0;
current_r = r0;

%初始化标志位为false
stop_signal = false;
for j = dt:dt:T
    %如果停止标志出现，立刻停止循环
    if stop_signal
        break;
    end
    t = round(j / dt);
    theta_head = current_theta - v_head * dt / current_r;
    r_head = p / (2 * pi) * theta_head;
    positions(1, :, t+1) = [r_head * cos(theta_head), ...
                r_head * sin(theta_head)];
    current_theta = theta_head;
    current_r = r_head;
    initial_theta = current_theta;
    initial_r = current_r;

    for i = 2:benches_num
        delta_theta(i) = L(i-1) / initial_r;
```

```
    initial_theta = initial_theta + delta_theta(i);
    initial_r = p / (2 * pi) * initial_theta;
    positions(i, 1, t+1) = initial_r * cos(initial_theta);
    positions(i, 2, t+1) = initial_r * sin(initial_theta);
end

if t > 0
    a = (positions(:, 1, t+1) - positions(:, 1, t)) / dt;  % vx
    b = (positions(:, 2, t+1) - positions(:, 2, t)) / dt;  % vy
    velocities(:, t+1) = sqrt(a.^2 + b.^2);
end


% %图像显示占用电脑资源影响运行速度，可以选择注释提高运行效率
% pause(0.01);
% clf;
% hold on;
% axis equal;
% xlabel('X (米)');
% ylabel('Y (米)');
% xlim([-12, 12]);
% ylim([-12, 12]);
% title(['板凳龙行进示意图 (t = ', num2str(j), 's)']);
% grid on;
% % 画背景螺线图
% theta_spiral = linspace(0, -32*pi, 10000);
% r_spiral = 0.55 * 16 + (0.55 / (2 * pi)) * theta_spiral;
% x_spiral = r_spiral .* cos(theta_spiral);
% y_spiral = r_spiral .* sin(theta_spiral);
% plot(x_spiral, y_spiral,'LineWidth', 2, 'Color', [0 0.4470 0.7410]);


%  key:考虑龙的宽度
%绘制出带有宽度的板凳龙
for i = 1:(benches_num-1)
    % 计算方向向量
    dx = positions(i+1, 1, t+1) - positions(i, 1, t+1);
    dy = positions(i+1, 2, t+1) - positions(i, 2, t+1);
    length = sqrt(dx^2 + dy^2);
    ux = -dy / length;  % 垂直方向的x分量
    uy = dx / length;   % 垂直方向的y分量

    % 计算每段线段两侧的四个顶点
    x_left1 = positions(i, 1, t+1) + ux * benches_width / 2;
    y_left1 = positions(i, 2, t+1) + uy * benches_width / 2;
    x_right1 = positions(i, 1, t+1) - ux * benches_width / 2;
    y_right1 = positions(i, 2, t+1) - uy * benches_width / 2;

    x_left2 = positions(i+1, 1, t+1) + ux * benches_width / 2;
    y_left2 = positions(i+1, 2, t+1) + uy * benches_width / 2;
    x_right2 = positions(i+1, 1, t+1) - ux * benches_width / 2;
    y_right2 = positions(i+1, 2, t+1) - uy * benches_width / 2;

    % 计算龙头方向向量
    dx = positions(2, 1, t+1) - positions(1, 1, t+1);
    dy = positions(2, 2, t+1) - positions(1, 2, t+1);
    length = sqrt(dx^2 + dy^2);
    ux = -dy / length;  % 垂直方向的x分量
```

```
    uy = dx / length;    % 垂直方向的y分量
    % 定义延长的倍数
    m = (3.41/2.86-1)/2;  % 延长1.5倍长度
    % 计算延长后的新的两个端点
    new_x1 = positions(1, 1, t+1) - m * dx;  % 起点向外延长
    new_y1 = positions(1, 2, t+1) - m * dy;
    new_x2 = positions(2, 1, t+1) + m * dx;  % 终点向外延长
    new_y2 = positions(2, 2, t+1) + m * dy;
    % 计算延长后的四个顶点
    new_x_left1 = new_x1 + ux * benches_width / 2;
    new_y_left1 = new_y1 + uy * benches_width / 2;
    new_x_right1 = new_x1 - ux * benches_width / 2;
    new_y_right1 = new_y1 - uy * benches_width / 2;

    %龙头的两个顶点
    P1_head=[new_x_left1,new_y_left1];  %龙头的起点
    P2_head=[new_x_right1,new_y_right1]; %龙头的终点

    %龙身的四个顶点
    Q1_body=[x_left1,y_left1];
    Q2_body=[x_right1,y_right1];
    Q3_body=[x_left2,y_left2];
    Q4_body=[x_right2,y_right2];

    new_x_left2 = new_x2 + ux * benches_width / 2;
    new_y_left2 = new_y2 + uy * benches_width / 2;
    new_x_right2 = new_x2 - ux * benches_width / 2;
    new_y_right2 = new_y2 - uy * benches_width / 2;

    if check_intersection(P1_head, P2_head, Q1_body, Q2_body) || ...
       check_intersection(P1_head, P2_head, Q2_body, Q4_body) || ...
       check_intersection(P1_head, P2_head, Q4_body, Q3_body) || ...
       check_intersection(P1_head, P2_head, Q3_body, Q1_body)

        disp(['龙头线段与龙身矩形相交，停止模拟 (t = ' num2str(j) 's)']);
        stop_signal = true;  % 设置标志
        break;
    end


    % %图像显示占用电脑资源影响运行速度，可以选择注释提高运行效率
    % % 使用 patch 函数绘制每段的宽线段
    % patch([x_left1, x_left2, x_right2, x_right1], ...
    % [y_left1, y_left2, y_right2, y_right1], 'b');


end

% %图像显示占用电脑资源影响运行速度，可以选择注释提高运行效率
% % 绘制延长后的宽线段
% patch([new_x_left1, new_x_left2, new_x_right2, new_x_right1], ...
%     [new_y_left1, new_y_left2, new_y_right2, new_y_right1], 'r');
% %画龙
% plot(positions(1, 1, t+1), positions(1, 2, t+1), 'ro-', ...
%     'MarkerSize', 0.3, 'LineWidth', 1,'MarkerFaceColor', 'r');
% plot(positions(2:end, 1, t+1), positions(2:end, 2, t+1), 'go-', ...
%     'MarkerSize', 3, 'LineWidth', 1,'MarkerFaceColor', 'r');
% hold off;
```

```
    disp(['当前运行' num2str(j) 's']);
end

%保存结果到Excel文件
writematrix(positions(:, 1,end),'result2.xlsx', 'Range', 'B2:B225');%x数据
writematrix(positions(:, 2,end),'result2.xlsx', 'Range', 'C2:C225');%y数据
writematrix(velocities(:,end),'result2.xlsx', 'Range', 'D2:D225');  %v数据
disp('数据已存入result2.xlsx');
```

```
%                                问题三：掉头空间                                %
%  (运行前请clear工作区,避免出现错误!)

% 参数设置
num_benches = 30;  % 只看30个板凳
head_length = 3.41;
body_length = 2.20;
benches_width = 0.3;
hole_to_head_distance = 0.275;
T=20;
v_head = 1.0;
dt = 0.01;
% 设置板凳孔的初始位置和角度设置
cnt=1; %计数

p=0.55;
dp=-0.0001;              %螺距每次减少0.01
r0 = 4.5;            % 初始时半径 4.5

stop_signal1 = false ;
stop_signal = false ;

for k =p:dp:0.3
    if stop_signal
        break;  % 退出主循环
    end
    stop_signal1 = false;

    cnt=cnt+1;
    theta0 = 9*pi/k;
    theta_start=theta0+pi;  %初始的角度增加180度
    r_start=theta_start*k/(2*pi);  %初始时刻龙头位置
    % 初始化龙头的位置
    positions(1,: , cnt) = [r_start * cos(theta_start), ...
        r_start * sin(theta_start)];
    L = [head_length - 2 * hole_to_head_distance;...
        repmat(body_length - 2 * hole_to_head_distance, num_benches-1, 1)];

    % 初始化龙身和龙尾位置
    initial_theta = theta_start;
    initial_r = r_start;
    for i = 2:num_benches
        delta_theta(i) = L(i-1) / initial_r;
        initial_theta = initial_theta + delta_theta(i);
        initial_r = k / (2 * pi) * initial_theta;
        positions(i, 1, cnt+1) = initial_r * cos(initial_theta);
        positions(i, 2, cnt+1) = initial_r * sin(initial_theta);
    end
        current_theta=theta_start;
        current_r = r_start;
    for j = dt:dt:T
        %出现终止信号退出循环
        if stop_signal1
            break;
        end
        t = round(j / dt);
        % 计算龙头位置
        theta_head = current_theta - v_head * dt / current_r;
```

```
r_head = k / (2 * pi) * theta_head;
positions(1, :, cnt+1) = [r_head * cos(theta_head), r_head * sin(theta_head)];

% 更新龙头前把手极坐标的角度和半径
current_theta = theta_head;
current_r = r_head;
% 更新第一节龙身前把手的角度和半径
initial_theta = current_theta;
initial_r = current_r;

% 计算龙身和龙尾位置
for i = 2:num_benches
    delta_theta(i) = L(i-1) / initial_r;
    initial_theta = initial_theta + delta_theta(i);
    initial_r = k / (2 * pi) * initial_theta;
    positions(i, 1, cnt+1) = initial_r * cos(initial_theta);
    positions(i, 2, cnt+1) = initial_r * sin(initial_theta);
end

 for i = 1:(num_benches-1)
    % 计算方向向量
    dx = positions(i+1, 1, cnt+1) - positions(i, 1, cnt+1);
    dy = positions(i+1, 2, cnt+1) - positions(i, 2, cnt+1);
    length = sqrt(dx^2 + dy^2);
    ux = -dy / length;  % 垂直方向的x分量
    uy = dx / length;   % 垂直方向的y分量

    % 计算每段线段两侧的四个顶点
    x_left1 = positions(i, 1, cnt+1) + ux * benches_width / 2;
    y_left1 = positions(i, 2, cnt+1) + uy * benches_width / 2;
    x_right1 = positions(i, 1, cnt+1) - ux * benches_width / 2;
    y_right1 = positions(i, 2, cnt+1) - uy * benches_width / 2;

    x_left2 = positions(i+1, 1, cnt+1) + ux * benches_width / 2;
    y_left2 = positions(i+1, 2, cnt+1) + uy * benches_width / 2;
    x_right2 = positions(i+1, 1, cnt+1) - ux * benches_width / 2;
    y_right2 = positions(i+1, 2, cnt+1) - uy * benches_width / 2;

    % 计算龙头方向向量
    dx = positions(2, 1, cnt+1) - positions(1, 1, cnt+1);
    dy = positions(2, 2, cnt+1) - positions(1, 2, cnt+1);
    length = sqrt(dx^2 + dy^2);
    ux = -dy / length;  % 垂直方向的x分量
    uy = dx / length;   % 垂直方向的y分量
    % 定义延长的倍数
    scale_factor = (3.41/2.86-1)/2;  % 延长1.5倍长度
    % 计算延长后的新的两个端点
    new_x1 = positions(1, 1, cnt+1) - scale_factor * dx;  % 起点向外延长
    new_y1 = positions(1, 2, cnt+1) - scale_factor * dy;
    new_x2 = positions(2, 1, cnt+1) + scale_factor * dx;  % 终点向外延长
    new_y2 = positions(2, 2, cnt+1) + scale_factor * dy;
    % 计算延长后的四个顶点
    new_x_left1 = new_x1 + ux * benches_width / 2;
    new_y_left1 = new_y1 + uy * benches_width / 2;
    new_x_right1 = new_x1 - ux * benches_width / 2;
    new_y_right1 = new_y1 - uy * benches_width / 2;

    new_x_left2 = new_x2 + ux * benches_width / 2;
```

```
    new_y_left2 = new_y2 + uy * benches_width / 2;
    new_x_right2 = new_x2 - ux * benches_width / 2;
    new_y_right2 = new_y2 - uy * benches_width / 2;


    % %图像显示占用电脑资源影响运行速度，可以选择注释提高运行效率
    % % 绘制延长后的宽线段
    % patch([x_left1, x_left2, x_right2, x_right1], ...
    %     [y_left1, y_left2, y_right2, y_right1], 'b');
    % patch([new_x_left1, new_x_left2, new_x_right2,...
    %     new_x_right1], [new_y_left1, new_y_left2, ...
    %     new_y_right2, new_y_right1], 'r');


    %龙头的线段
    cnt1_head=[new_x_left1,new_y_left1];  %龙头的起点
    cnt2_head=[new_x_right1,new_y_right1]; %龙头的终点

    %龙身的矩形
    Q1_body=[x_left1,y_left1];
    Q2_body=[x_right1,y_right1];
    Q3_body=[x_left2,y_left2];
    Q4_body=[x_right2,y_right2];

    if check_intersection(cnt1_head, cnt2_head, Q1_body, Q2_body) || ...
       check_intersection(cnt1_head, cnt2_head, Q2_body, Q4_body) || ...
       check_intersection(cnt1_head, cnt2_head, Q4_body, Q3_body) || ...
       check_intersection(cnt1_head, cnt2_head, Q3_body, Q1_body)
        stop_signal1 = true;  % 设置标志
    end
end
if sqrt(positions(1, 1, cnt+1).^2+positions(1, 2, cnt+1).^2)<4.5
    stop_signal = false;
else
    stop_signal = true;
end

disp(['p = ' num2str(k) 'm)']);
disp(['t = ' num2str(j) 's)']);


% %图像显示占用电脑资源影响运行速度，可以选择注释提高运行效率
% pause(0.01);
% clf;
% hold on;
% axis equal;
% xlabel('X (米)');
% ylabel('Y (米)');
% xlim([-12, 12]);
% ylim([-12, 12]);
% grid on;
% title(['板凳把手位置示意图 (p = ', num2str(k), 'm)(t = ',...
%        num2str(j), 's)']);
% plot(positions(1:end, 1, cnt+1), positions(1:end, 2, cnt+1), ...
%     'go-', 'MarkerSize', 3, 'LineWidth', 1,'MarkerFaceColor', 'r');
% hold off;
```

```
    end

end
    disp(['龙头线段与龙身相交，停止模拟 (p = ' num2str(k-dp) 'm)']);
```

```
%                          问题四:最小掉头路径                          %

% 计算 4.5/(1.7/(2*pi)) 的结果
theta=4.5/(1.7/(2*pi));
ro=4.5;
dy=(1.7/(2*pi))*sin(theta)+(1.7/(2*pi))*theta*cos(theta);
dx=(1.7/(2*pi))*cos(theta)-(1.7/(2*pi))*theta*sin(theta);
kk=dy/dx;
k=-1/kk;
y=ro*sin(theta);
x=ro*cos(theta);

% 计算直线方程的参数
bb= y - kk*x;
b = y - k * x; % 直线方程为 y = kx + b，b为截距

% 计算直线到原点的距离
dd = abs(bb) / sqrt(1 + kk^2);  %切线
d = abs(b) / sqrt(1 + k^2);

% 显示结果
disp(['垂线距离为: ', num2str(d)]);
disp(['切线距离为: ', num2str(dd)]);

% 定义目标函数和约束条件
objective = @(a) 2*d*a./sin(pi-a);
constraint = @(a) (2*d*a./sin(a)).*(1 + cos(pi-a))./a - 2*dd;

% 定义范围
a_min = pi/2;
a_max = pi;

% 设置优化选项
options = optimoptions('fmincon','Display','off');

% 初始猜测
a0 = (a_min + a_max) / 2;

% 使用 fmincon 进行优化
[a_opt, s_opt] = fmincon(@(a) 2*d*a./sin(pi-a), a0, [], [], [], [], ...
    a_min, a_max, @(a) deal([], constraint(a)), options);

% 输出结果
disp(['s最小值为: ', num2str(s_opt)]);
disp(['对应的phi值为: ', num2str(a_opt)]);

%求题目给出比例的圆弧长
syms l theta0
assume(theta0 > pi/2 & theta0 < pi);
eq1 = l*(1+(cos(pi-theta0))) == 2*dd;
eq2 = l*sin(pi-theta0) == 2*d;
sol = solve([eq1, eq2], [l, theta0]);
x_sol = sol.l;
y_sol = sol.theta0;
fprintf('题设条件下 m = %.4f\n', x_sol);
fprintf('题设条件下 phi = %.4f\n', y_sol);
fprintf('题设条件下 s = %.4f\n', x_sol*y_sol);
```