

EE 5325/4315 – Kinematics of Mobile Robots, Summer 2004  
Jose Mireles Jr.,

The lower MATLAB functions (uncompleted on purpose) display three graphs showing the control strategies for driving a Three wheeled mobile robot from position (0,0) to position (10,10), having obstacles in 2D coordinate positions (3,0), and (7,9).

```
function xdot=NewPathMob(tspan,X);

%%% COPY FROM HERE
%Obstacles:
X1 = 3; Y1 = 0;
X2 = 7; Y2 = 9;
%
%Goal:
GoalX= 10; Goaly=10;
%
% T=linspace(0,25,1000);
% [t,X]=ode23('NewPathMob',T,[0 0 -pi/2 0]); %pointing towards one of the
% obstacles
% figure(1), plot(X(:,1),X(:,2))
% title('Path planning and Control of Mobile Robots')
% xlabel('Obstacles at "o" marks, and goal at "x" mark')
% hold on, plot (X1,Y1,'o'), plot (X2,Y2,'o'), plot (GoalX,Goaly,'x')
% grid on
% figure(2), plot(X(:,3)*180/pi), title('Phi')
% figure(3), plot(X(:,4)*180/pi), title('Alpha')
%
%%% TO HERE to MATLAB Window !!!

% Specification of system (THREE-WHEELED MOBILE ROBOT)
% X = { X(0) = x; X(1) = y; X(3) = Phi; X(4) = alpha }

L = 0.3; % 30 cms.
k1 = __; k2 = __; k3 = __; % GAINS NOT SHOWN ON PURPOSE!
kp = __; % GAIN NOT SHOWN ON PURPOSE!
kv = __; % Error coefficient for velocity NOT SHOWN ON PURPOSE!

% Calculating the Potential vectors:
% Obstacle1 located at (3,0)
Mag1x = (X(1)-X1);
Mag1y = (X(2)-Y1);
M1 = sqrt(Mag1x^2+Mag1y^2);
if M1==0 M1=0.00001; end %due to the division below
% Obstacle2 located at (7,10)
Mag2x = (X(1)-X2);
Mag2y = (X(2)-Y2);
M2 = sqrt(Mag2x^2+Mag2y^2);
if M2==0 M2=0.00001; end %due to the division below
% Goal located at (10,10)
Mag3x = (X(1)-GoalX);
Mag3y = (X(2)-Goaly);
M3 = sqrt(Mag3x^2+Mag3y^2);
if M3==0 M3=-0.00001; end %due to the division below

Fx = k1*Mag1x/M1 + k2*Mag2x/M2 - k3*Mag3x/M3;
Fy = k1*Mag1y/M1 + k2*Mag2y/M2 - k3*Mag3y/M3;
PhiD = atan2(-Fx,Fy);

% The distance of the desired curvature vector is the magnitude of the
% reaction force
R = sqrt(Fx^2+Fy^2);

% Velocity of the Wheel depends on the magnitud of the reaction/action force
Vt = R/kv;

% Tracking Error in Phi (the direction of Phi is negative!)
Perror = kp * PhiD-X(3);
if (Perror>pi/2)
    Perror = Perror-pi/4;
end
if (Perror<-pi/2)
    Perror = Perror+pi/4;
end

% Calculating Control angle Alpha:
alpha = _____ + atan2(L,R); % Alpha depends on Perror
if (alpha>pi/2)
    alpha = alpha-pi/4;
end
if (alpha<-pi/2)
    alpha = alpha+pi/4;
end

xdot=[
    Vt*(-sin(X(3)))*cos(alpha); % Driven by desired alpha and Vt velocity
    Vt*cos(X(3))*cos(alpha); % Driven by desired alpha and Vt velocity
    Vt*sin(alpha)/L; % Driven by desired alpha and Vt velocity
    (alpha-X(4))*25/1000; % Due that each simulation runs for 25/1000 seconds
];
```

