UNIVERSITY AT BUFFALO

MAE513

HW6-Wheeled Mobile Manipulator

Hrishi Shah
Sourish Chakravarty
4/7/2009

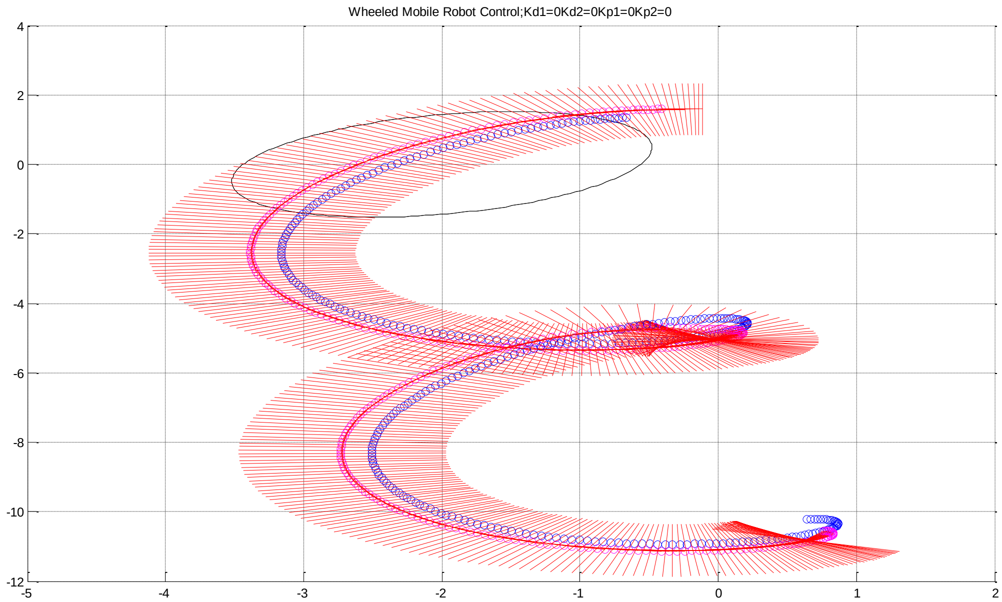
In this homework, we try to study the control of a wheeled mobile robot, a system with a non-holonomic constraint. We develop here a kinematic analysis method and look at its performance as the control parameters are changed.

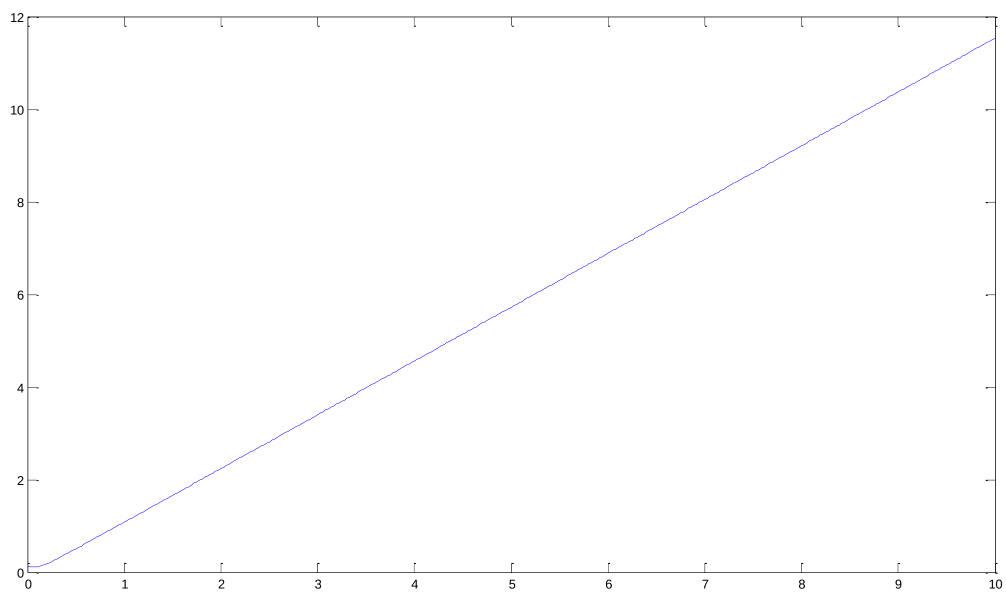
Table of Contents

1) Only Position Level Control

a. $K_p=0$, $K_d=0$

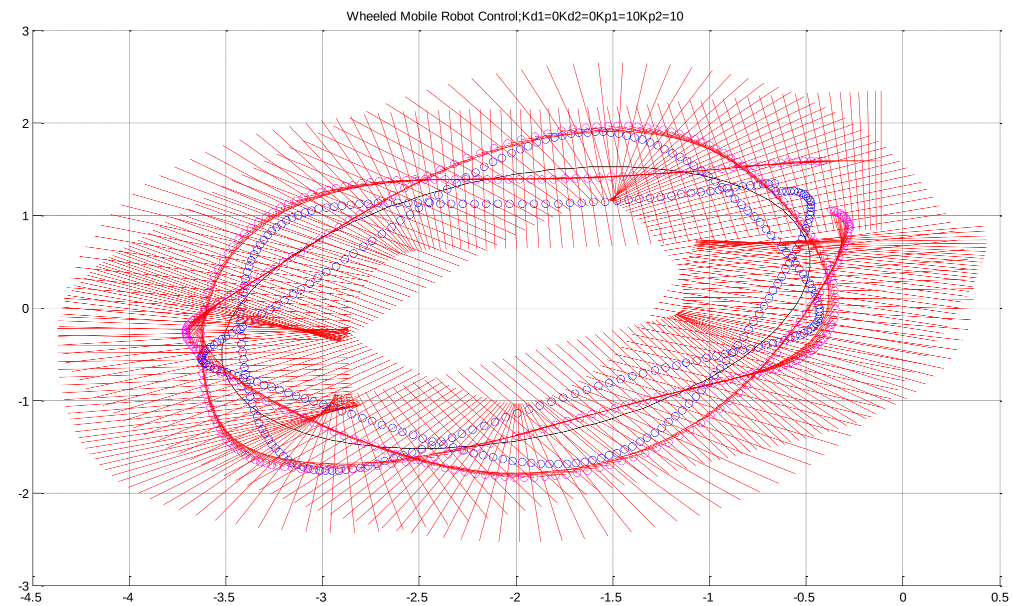
As we can see, the system is unstable and error is increasing linearly. This is due to the absence of control.

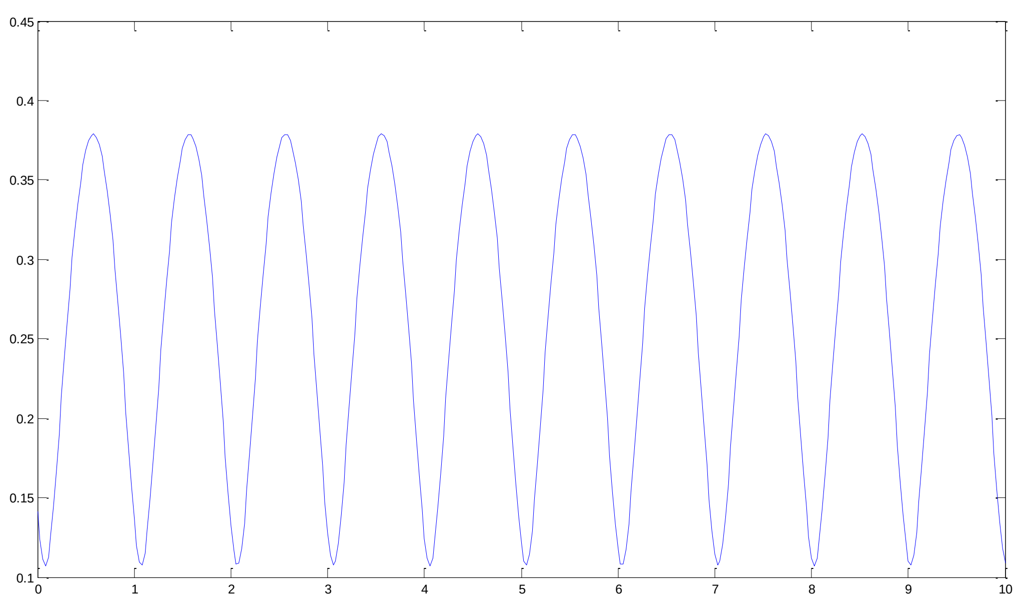




b. K_p=10, K_d=0

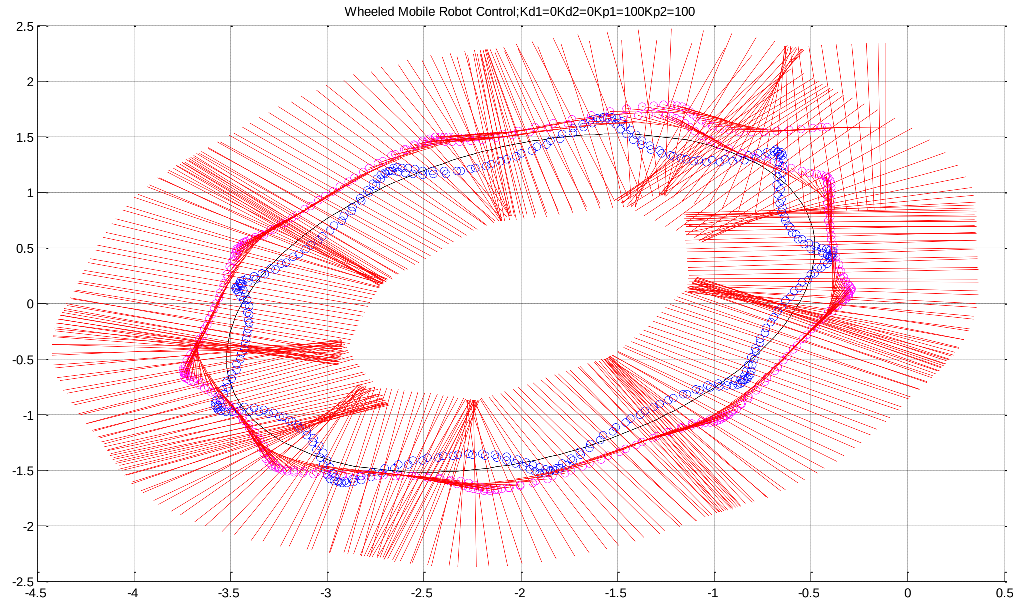
In this case, we provide a low control on the displacement error. So, the output fluctuates around the desired input at a low frequency. This is easily seen in the error plot.

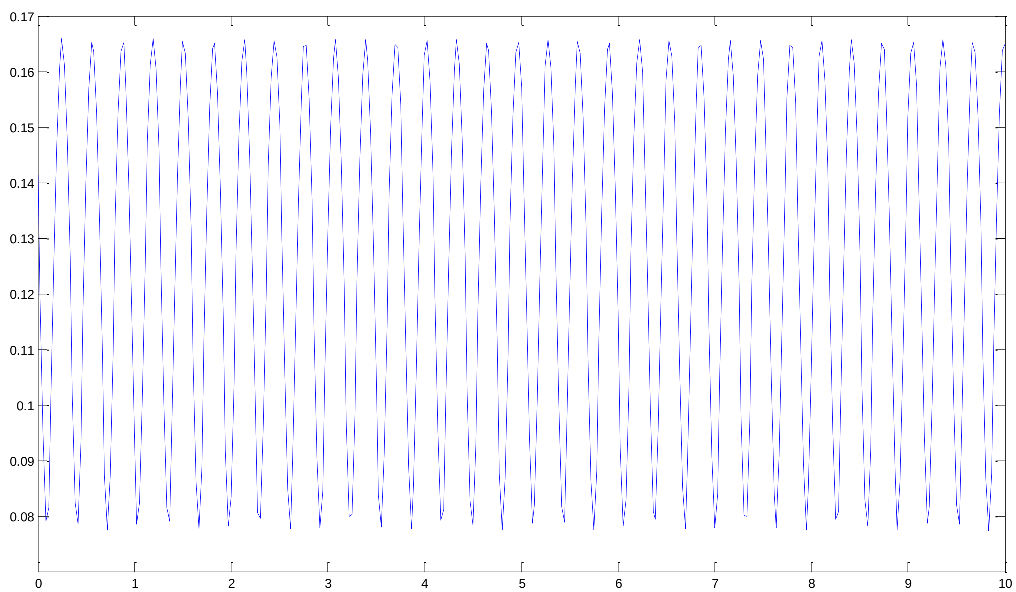




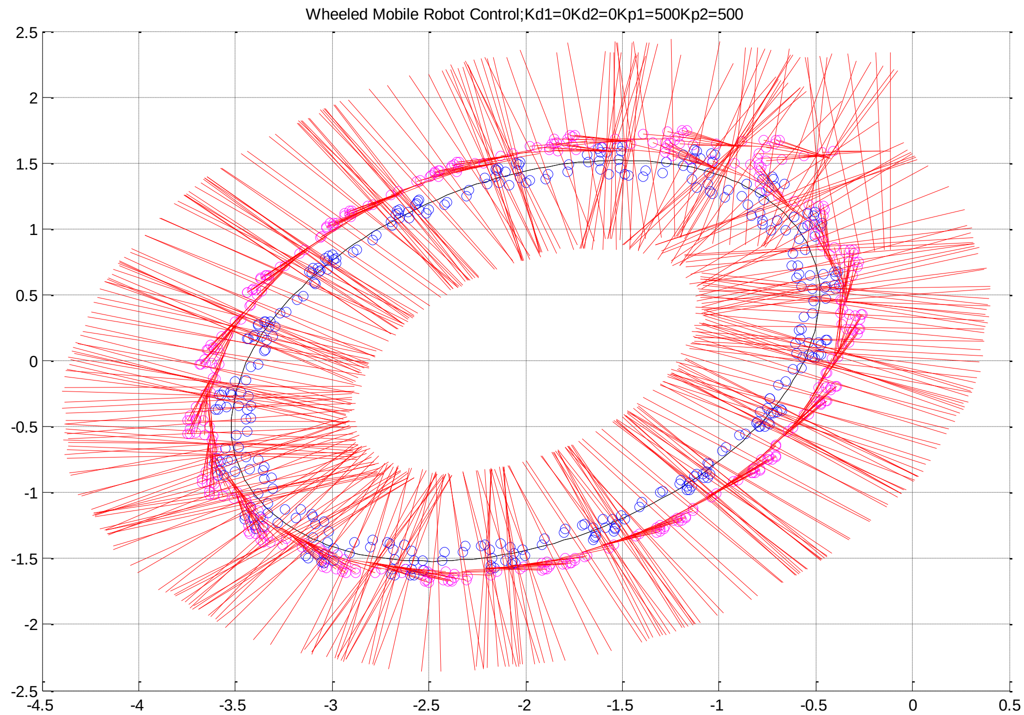
c. Kp=100, Kd=0

Now, on increasing the control on the displacement error, we find an increased frequency of oscillation and consecutively a decreased amplitude of the error.

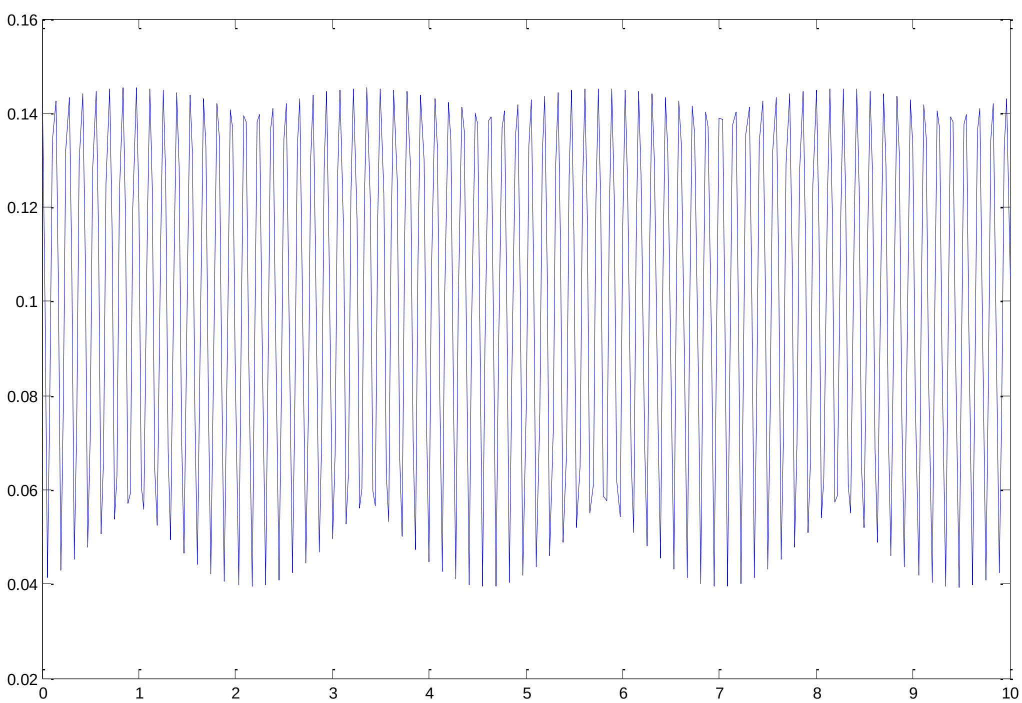




d. K_p=500, K_d=0



Wheeled Mobile Robot Control;Kd1=0Kd2=0Kp1=500Kp2=500
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-4.5
-4
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5

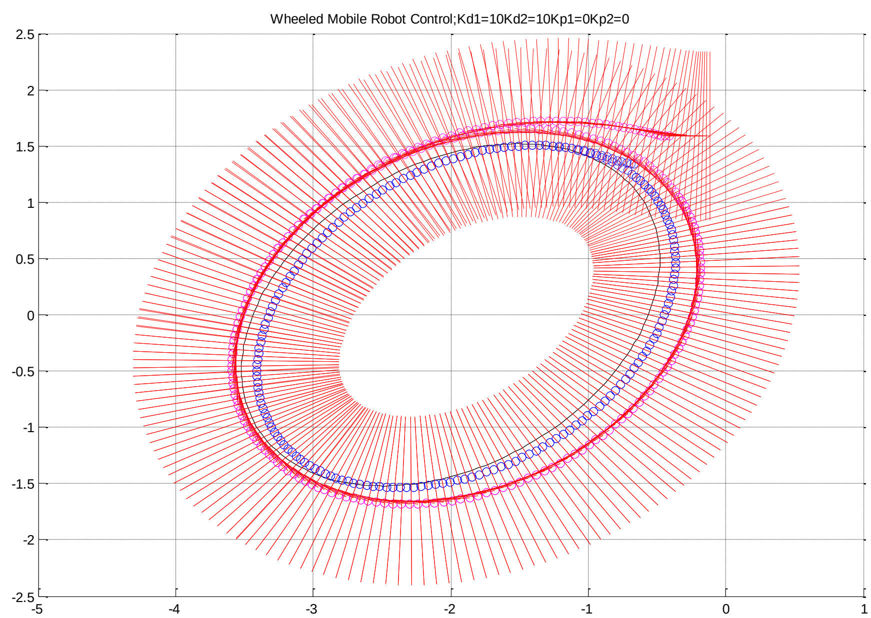


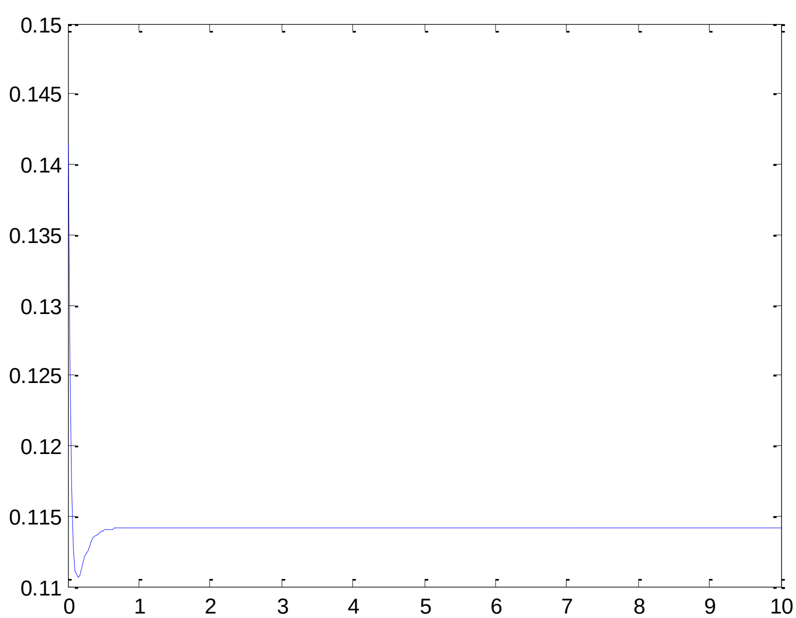
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1
2
3
4
5
6
7
8
9
10

2) Only Velocity Level Control

a. K_p=0, K_d=10

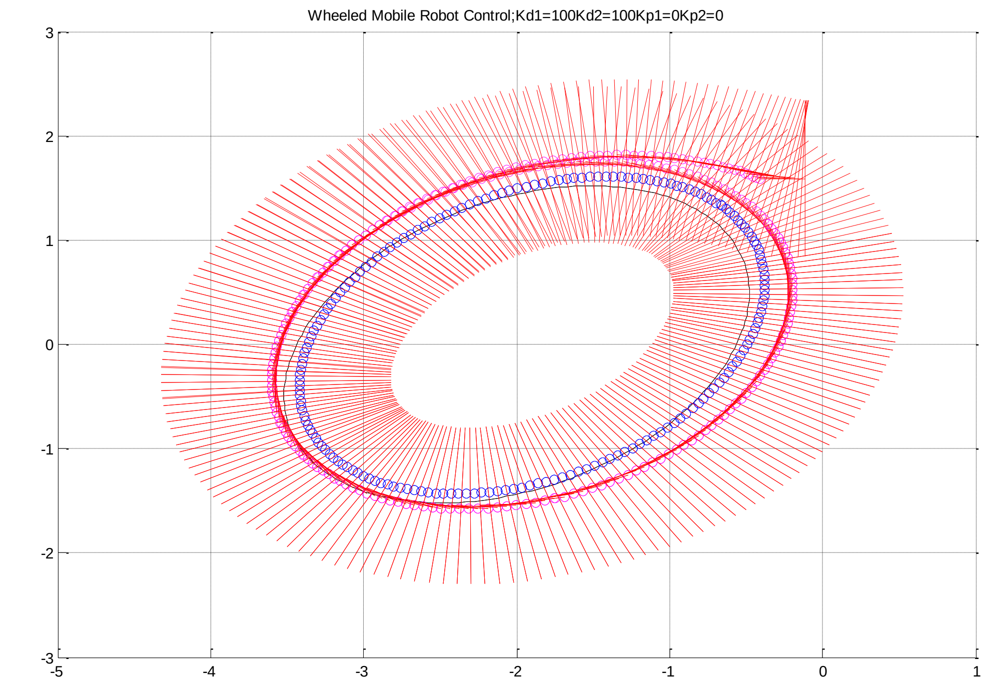
We investigate herein the effect of having only a velocity based controller. The least bit of control reduces the error by a great amount. The steady state error remains because of the absence of a position control term.

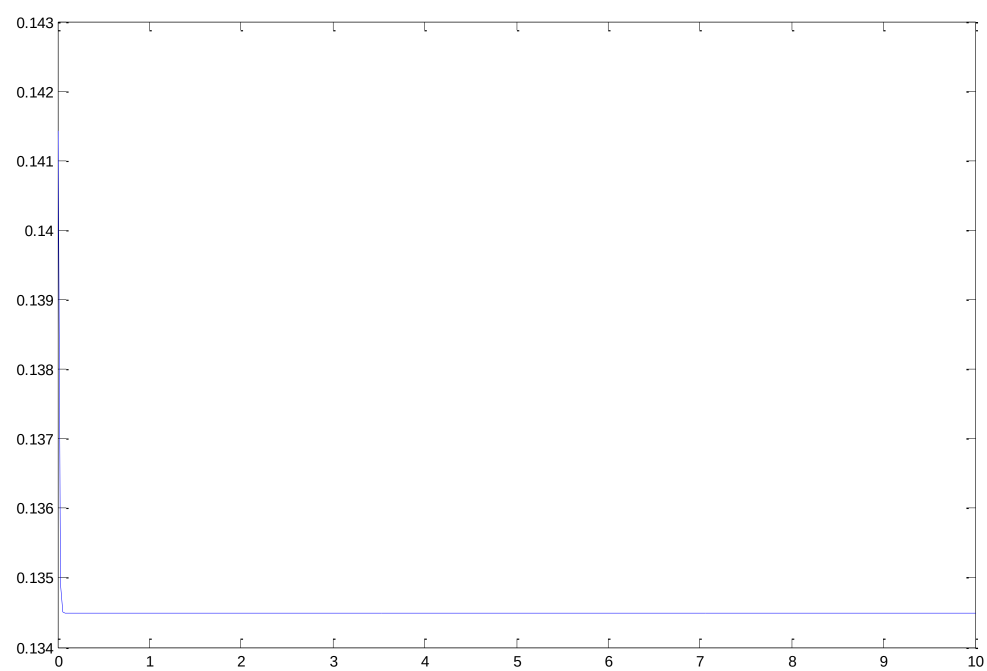




b. K_p=0, K_d=100

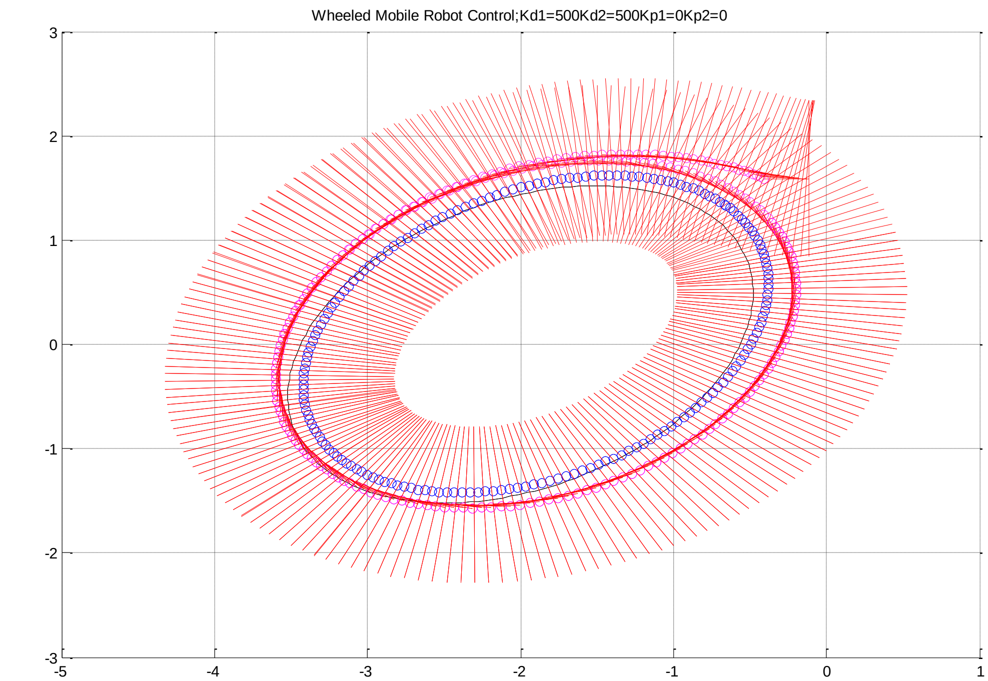
As can be seen, the convergence is much faster and the overshoot is almost neutralized. However, the steady state error prevails since position control is still off.

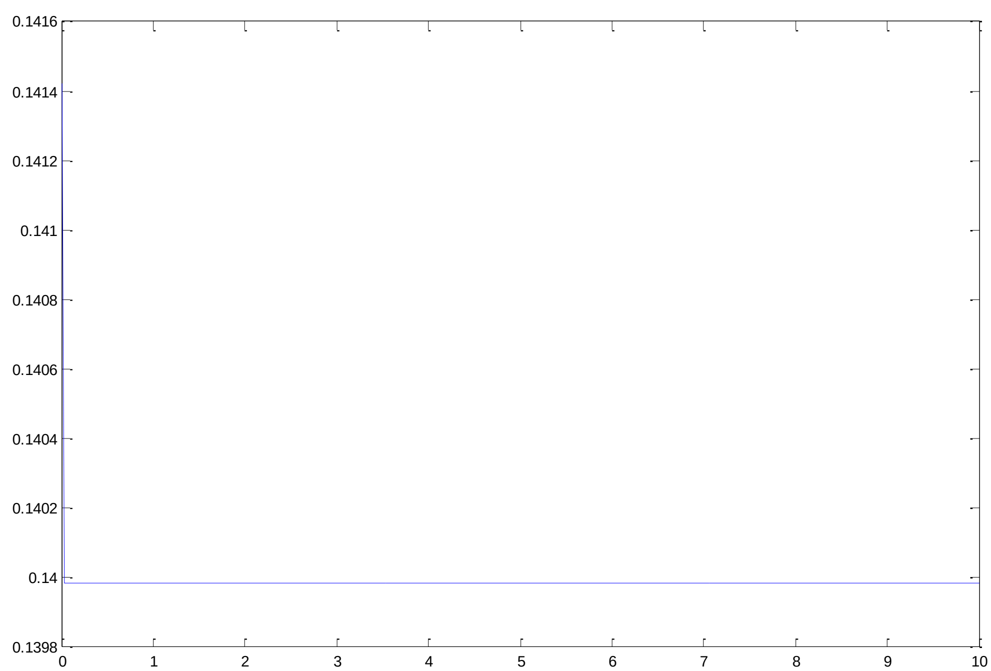




c. K_p=0,K_d=500

Now, we observe that increasing the velocity control no more improves the performance of this system. So, we stop at this point and start including both velocity and position level control.

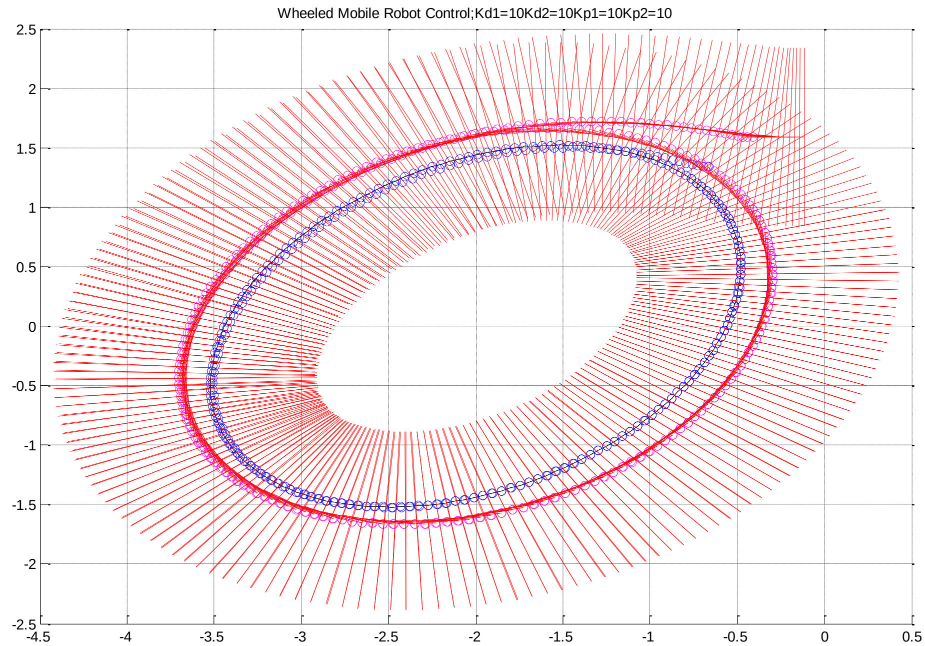


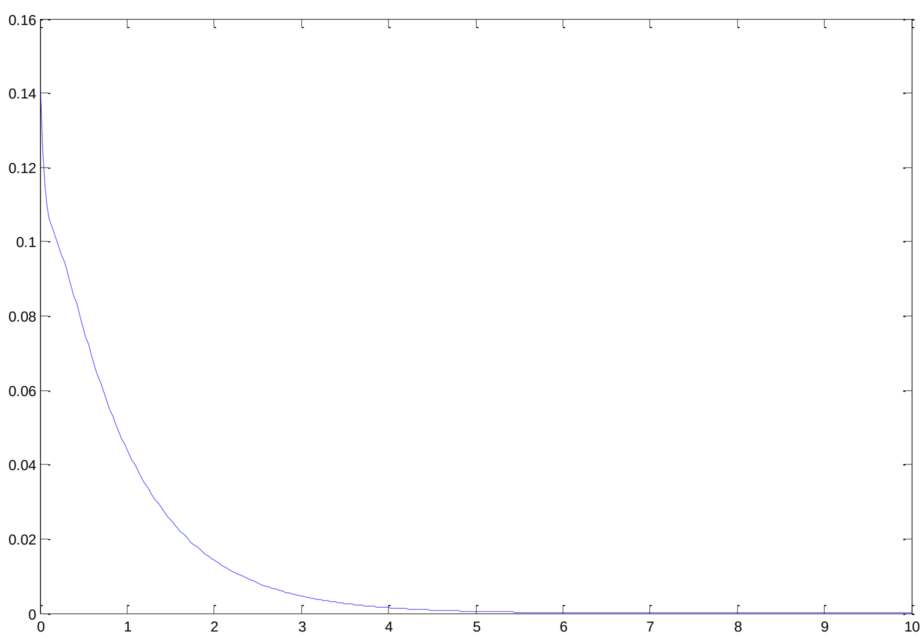


3) Both Position and Velocity Level Control

a. K_p=10(low),K_d=10(low)

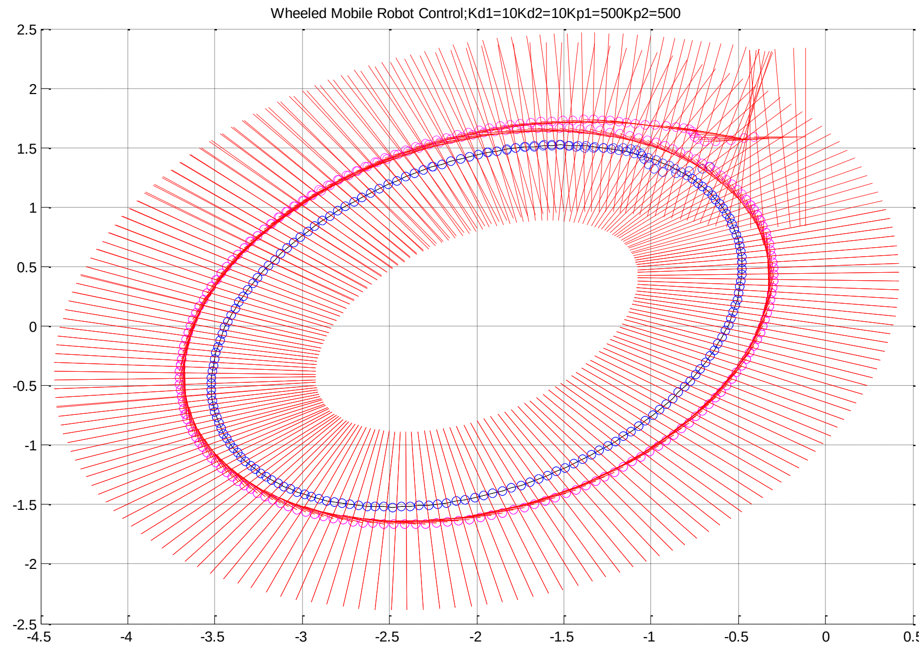
We see that although the error is eventually reduced to zero, the system response is not at all desirable since it oscillates a bit and also has a high settling time.

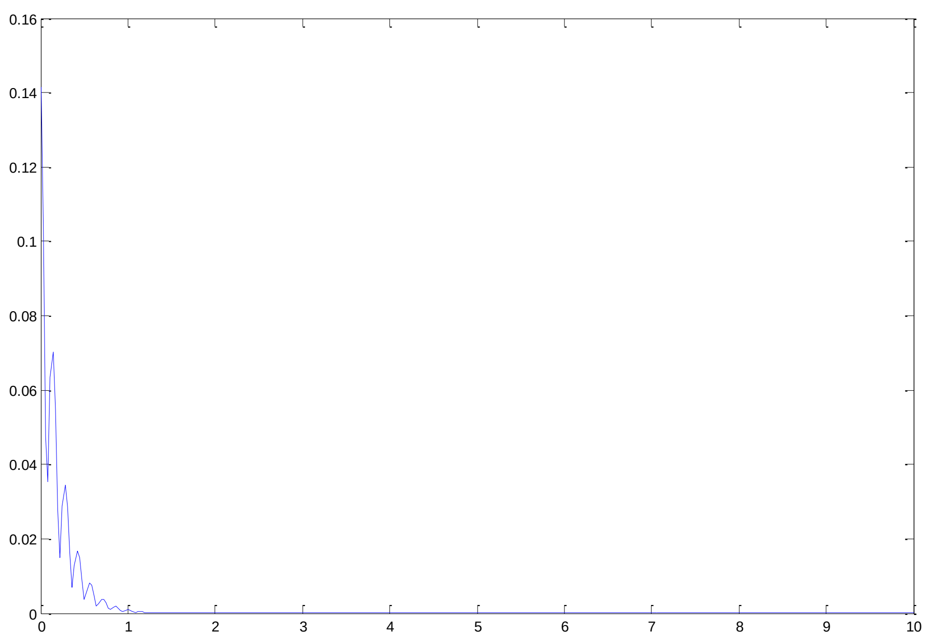




b. K_p=500(high), K_d=10(low)

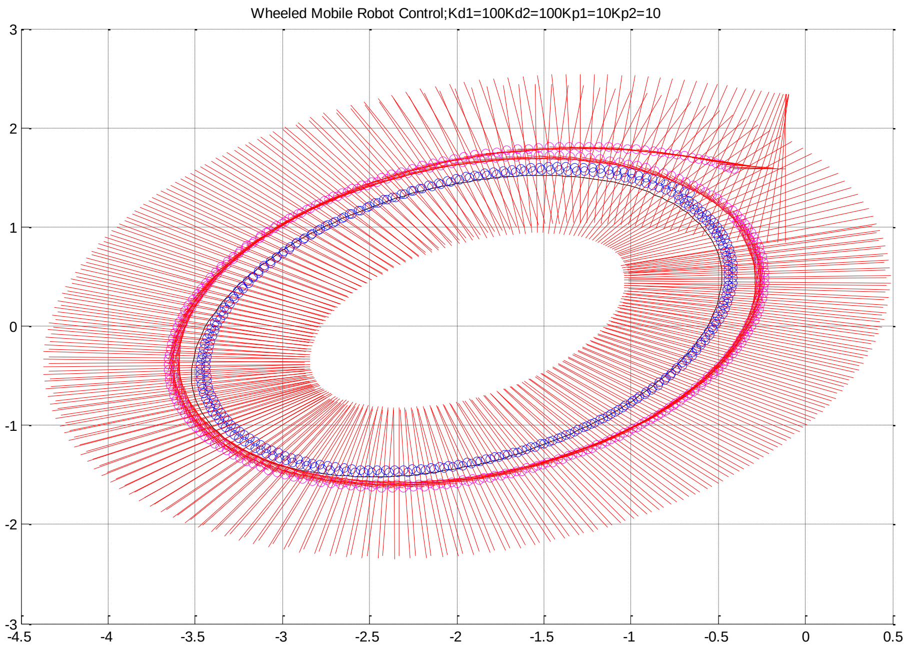
Now the error fluctuates a bit but eventually boils down to zero. This may be acceptable for poor expectations on system response to disturbances.

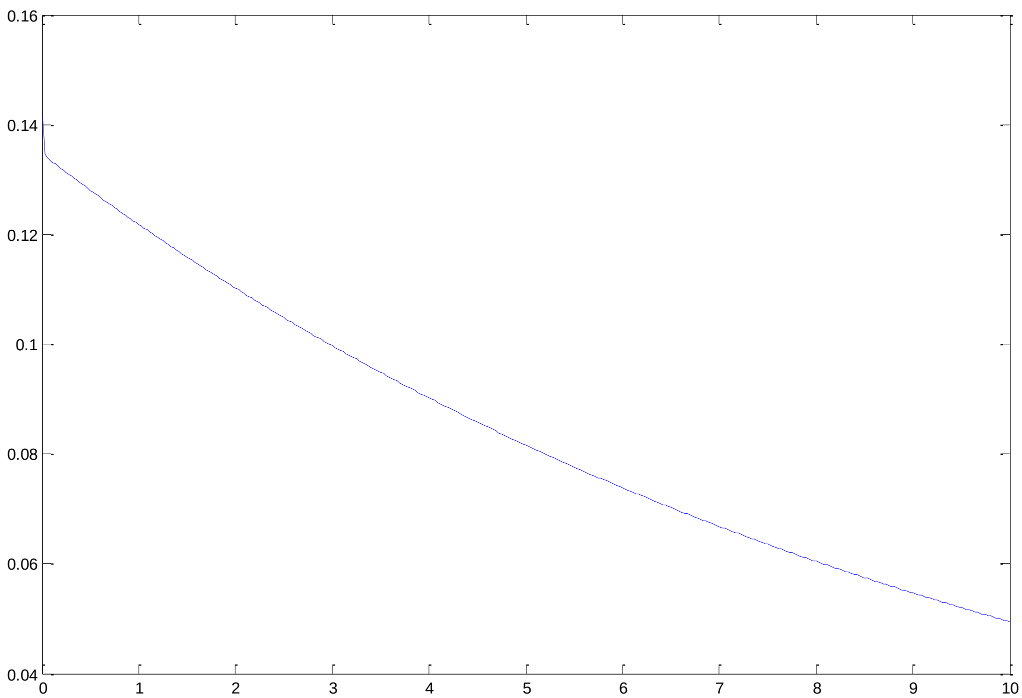




c. K_p=10(low), K_d=100(high)

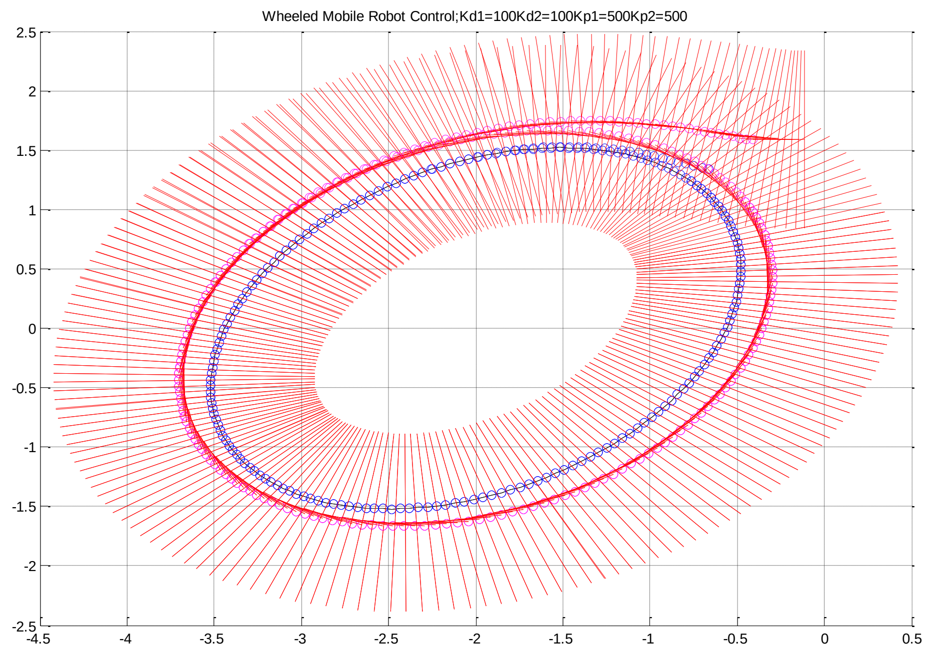
In this case, the oscillatory behavior is suppressed fairly well but it makes the settling time unacceptable. So, this combination is not at all favorable.

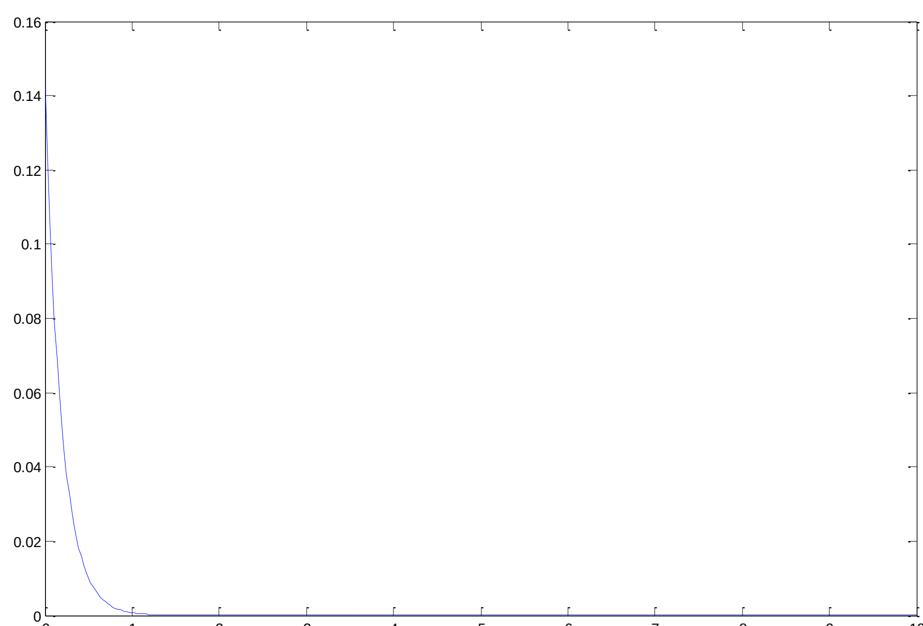




d. K_p=500(high), K_d=100(high)

In this case, the oscillatory behavior is fairly well suppressed even though the convergence is about the same as case b (settling time is about 0.1s for both this and the case b). This is a very well conditioned response and should satisfy most design requirements.





4) Conclusion

We were able to study and condition the response of a Wheeled Mobile Robot Manipulator quite comfortably. This was mainly due to the formulation that now made the error dynamic to resemble a second order system.

We noticed three main things:

a) Only velocity level control eliminates oscillations but leaves a steady state error.
b) Only position level control leads to an oscillatory error that does not decay with time.
c) Applying both position and velocity level control leads the system to behave like a typical second order system, and this allows us to "condition" the error as we desire.

5) Appendix

a. Main file

```
clear all;
close all;
clc;
global r b d a mc mw Ic Iw Im L1 L2 % WMR paramters
global xe ye rx ry ell_an start_an w % Trajectory information

d2r=pi/180;

%% SYSTEM PARAMETERS OF WMR
r= 0.15; % Radius of each wheel
b= 0.75;
d= 0.30;
a= 2.00;
mc= 30.00;
mw= 1.00;
Ic= 15.625;
Iw= 0.005;
Im= 0.0025;
L1 = 0.25; L2 =0.25; % Cood of Look Ahead point in WMR ref frame

% TRAJECTORY (ELLIPSE) INFORMATION
xe=-2; % x center of ellipse
ye=0; % y center of ellipse
rx=1.75;% Half-length of major axis
ry=1.25; % Half-length of minor axis
ell_an=45*d2r; % Angle subtended between major axis and horizontal
start_an= 0*d2r; % Initial phase at t=0
w=72*d2r;

% INITIAL CONDITIONS
t=0;
% Position of look ahead point
x_E0=xe+rx*cos(w*t + start_an)*cos(ell_an) + ry*sin(w*t + start_an)*(-
sin(ell_an)); % Initial X on ellipse
y_E0=ye+rx*cos(w*t + start_an)*sin(ell_an) + ry*sin(w*t +
start_an)*(cos(ell_an)); % Initial Y on ellipse
```

```
% xd_0= w*( -rx*sin(w*t + start_an)*cos(ell_an) + ry*cos(w*t +
start_an)*(-sin(ell_an)) ); % Initial X-vel on ellipse
% yd_0= w*( -rx*sin(w*t + start_an)*sin(ell_an) + ry*cos(w*t +
start_an)*(cos(ell_an)) ); % Initial Y-vel on ellipse

% x_0=xe+rx*cos(ell_an); % Initial X of CM of WMR
% y_0=ye+rx*sin(ell_an); % Initial Y of End Effector
phi_0 = 180*d2r;
v= 1; % Linear velocity of CM
om= 5*d2r; % Ang velocity of WMR about CM
th_R_0= 0*d2r;
th_L_0= 0*d2r;

TR_C2O= [cos(phi_0), -sin(phi_0);
        sin(phi_0), cos(phi_0)]      % Transformation from CM cood system
to Absolute cood sys
X_C_0= [x_E0,y_E0]' - TR_C2O*[L1,L2]'; % Initial position of CM
x_0= X_C_0(1)+.1;
y_0= X_C_0(2)+.1;
c= r/(2*b);
S= [c*(b*cos(phi_0)-d*sin(phi_0)), c*(b*cos(phi_0)+d*sin(phi_0));
    c*(b*sin(phi_0) + d*cos(phi_0)),c*(b*sin(phi_0) - d*cos(phi_0));
    c,-c];
Nu= pinv(S)*([v*cos(phi_0), v*sin(phi_0), om]');
thd_R_0= Nu(1);
thd_L_0= Nu(2);

% SIMULATION PARAMETERS
n=360; % total number of points in simulation
sim_time=10;% 2*pi/wc; % total simulation time
dt=sim_time/n;
tspan=0:dt:sim_time;
% tspan=[0,10];

SIMU=1;
switch(SIMU)
    case 1 % Simulation 1
        global Kp1 Kp2 Kd1 Kd2
        Kp1=10;  Kp2=Kp1;
        Kd1=100;   Kd2=Kd1;

        options= odeset('RelTol',1e-4,'AbsTol',[1e-6, 1e-6, 1e-4, 1e-4,
1e-4,1e-4, 1e-4]);
        X0=[x_0, y_0, phi_0, th_R_0, th_L_0, thd_R_0, thd_L_0];
        [t,Y]=ode45(@WMR_TRAJTRACK,tspan,X0,options);
        txt1=['Wheeled Mobile Robot
Control;Kd1=',num2str(Kd1),'Kd2=',num2str(Kd2),'Kp1=',num2str(Kp1),'Kp2=',
num2str(Kp2)];
        plotbot_WMR(t,Y,1,txt1)
    otherwise
        disp('incorrect TRIG value');
end
```

b. Function Call file

```
function [dX]= WMR_TRAJTRACK(t,X)
```

```
global b d a mc mw Ic Iw Im r L1 L2% WMR paramters
global xe ye rx ry ell_an start_an w % Trajectory information
global Kp1 Kp2 Kd1 Kd2

t
%% CURRENT STATE INFORMATION
x_C= X(1);
y_C= X(2);
phi= X(3);
th_R= X(4);
th_L= X(5);
thd_R= X(6);
thd_L= X(7);

c= r/(2*b);
S= [c*(b*cos(phi)-d*sin(phi)), c*(b*cos(phi)+d*sin(phi));
    c*(b*sin(phi) + d*cos(phi)),c*(b*sin(phi) - d*cos(phi));
    c,-c;
    1,0;
    0,1];


Nu= [thd_R,thd_L]'; % Minimal coordinate

Q_dot = S*Nu; % Extended coordinate
xd_C= Q_dot(1);
yd_C=Q_dot(2);
om= Q_dot(3); % Phi_dot

Sd = [c*om*(-b*sin(phi)-d*cos(phi)), c*om*(-b*sin(phi)+d*cos(phi));
    c*om*(b*cos(phi) - d*sin(phi)),c*om*(b*cos(phi) + d*sin(phi));
    0,0;
    0,0;
    0,0];


TR_C2O= [cos(phi), -sin(phi);
        sin(phi), cos(phi)]; % Transformation from CM cood system to
Absolute cood sys
%% ELLIPSE INFORMATION : DESIRED OUTPUTS

x_E=xe+rx*cos(w*t + start_an)*cos(ell_an) + ry*sin(w*t + start_an)*(-
sin(ell_an)); % Initial X on ellipse
y_E=ye+rx*cos(w*t + start_an)*sin(ell_an) + ry*sin(w*t +
start_an)*(cos(ell_an)); % Initial Y on ellipse

xd_E= w*( -rx*sin(w*t + start_an)*cos(ell_an) + ry*cos(w*t + start_an)*(-
sin(ell_an)) ); % Initial X-vel on ellipse
yd_E= w*( -rx*sin(w*t + start_an)*sin(ell_an) + ry*cos(w*t +
start_an)*(cos(ell_an)) ); % Initial Y-vel on ellipse

xdd_E= w*w*( -rx*cos(w*t + start_an)*cos(ell_an) - ry*sin(w*t +
start_an)*(-sin(ell_an)) ); % Initial X-acc on ellipse
ydd_E= w*w*( -rx*cos(w*t + start_an)*sin(ell_an) - ry*sin(w*t +
start_an)*(cos(ell_an)) ); % Initial Y-acc on ellipse
```

```
%% ERROR MODELING

X_C_O= [x_C,y_C]';
X_L_C= TR_C2O*[L1,L2]';
X_L_O= X_C_O+X_L_C; % Position of Look Ahead point in {O} frame= Xco + Xlc
x_L=X_L_O(1);
y_L=X_L_O(2);

Xd_C_O= [xd_C,yd_C]'; % Velocity of Look Ahead point in {O} frame
Xd_L_C= (om*[L1,L2]*[0,1;-1,0]*TR_C2O')';
Xd_L_O= Xd_C_O + Xd_L_C;
xd_L=Xd_L_O(1);
yd_L=Xd_L_O(2);

xdd_L = xdd_E - Kp1*(x_L - x_E) - Kd1*(xd_L - xd_E); % X-Acc of Look Ahead
point
ydd_L = ydd_E - Kp2*(y_L - y_E) - Kd2*(yd_L - yd_E); % Y-Acc of Look Ahead
point

%% INPUT MODELING
% J= [1,0,-L1*sin(phi)-L2*cos(phi),0,0;
%     0,1,L1*cos(phi)-L2*sin(phi),0,0];
% Ptemp=J*S
p11= c*((b-L2)*cos(phi) - (d+L1)*sin(phi));
p12= c*((b+L2)*cos(phi) + (d+L1)*sin(phi));
p21= c*((b-L2)*sin(phi) + (d+L1)*cos(phi));
p22= c*((b+L2)*sin(phi) - (d+L1)*cos(phi));
P=[p11,p12;
    p21,p22];

det(P);
pd11= c*(Nu(1) - Nu(2))*c*(-(b-L2)*sin(phi) - (d+L1)*cos(phi));
pd12= c*(Nu(1) - Nu(2))*c*(-(b+L2)*sin(phi) + (d+L1)*cos(phi));
pd21= c*(Nu(1) - Nu(2))*c*( (b-L2)*cos(phi) - (d+L1)*sin(phi));
pd22= c*(Nu(1) - Nu(2))*c*( (b+L2)*cos(phi) + (d+L1)*sin(phi));
Pd= [pd11,pd12;
    pd21,pd22];

v=[xdd_L, ydd_L]';
u = inv(P)*(v - Pd*Nu);

%% Dynamic Modeling
m= mc+ 2*mw;
I = Ic + 2*mw*(d^2 + b^2) + 2*Im;

M = [m, 0, 2*mw*d*sin(phi), 0 , 0;
    0, m, -2*mw*d*cos(phi), 0, 0;
    2*mw*d*sin(phi), -2*mw*d*cos(phi), I, 0, 0;
    0, 0, 0, Iw, 0;
    0, 0, 0, 0, Iw];

V = [2*mw*d*(om^2)*cos(phi);
    2*mw*d*(om^2)*sin(phi);
    0;
    0;
    0];
```

```
E = [0, 0;
    0, 0;
    0, 0;
    1, 0;
    0, 1];


f2= inv(S'*M*S)*(-S'*M*Sd*Nu - S'*V);
% size(inv(S'*M*S)*S'*E)
Tau= pinv(inv(S'*M*S)*S'*E)*(u-f2);
% Tau2= (S'*M*S)*u + S'*M*Sd*Nu + S'*V
%
% [S*Nu; f2]
% [zeros(5,2); inv(S'*M*S)*S'*E]*Tau;
dX = [S*Nu; f2] + [zeros(5,2); inv(S'*M*S)*S'*E]*Tau;
% pause
```

c. Plotbot file

```
function plotbot_WMR(t,X,index,txt1)
global r b d a mc mw Ic Iw Im L1 L2 % WMR paramters
global xe ye rx ry ell_an start_an w % Trajectory information
% aviobj = avifile([txt1,'.avi'],'compression','Cinepak'); % Declare an
avi object

figure(index*3-2);
cla('reset');
% axis manual;
% axis([-3 5 -3 3]);
hold on;
grid on;
% T=[0:0.1:20]
x_E=xe+rx*cos(w*t + start_an)*cos(ell_an) + ry*sin(w*t + start_an)*(-
sin(ell_an)); % Initial X on ellipse
y_E=ye+rx*cos(w*t + start_an)*sin(ell_an) + ry*sin(w*t +
start_an)*(cos(ell_an)); % Initial Y on ellipse
plot(x_E,y_E,'-k');
% plot(xc+rc*cos(wc*t+start_an),yc+rc*sin(wc*t+start_an),'-k');
title(txt1);
for i=1:length(t)
    x_C=X(i,1);
    y_C=X(i,2);
    phi=X(i,3);
    thR=X(i,4);
    thL=X(i,5);

    TR_C2O= [cos(phi), -sin(phi);
        sin(phi), cos(phi)]; % Transformation from CM cood system to
Absolute cood sys

    %% ELLIPSE INFORMATION : DESIRED OUTPUTS

    x_E=xe+rx*cos(w*t(i) + start_an)*cos(ell_an) + ry*sin(w*t(i) +
start_an)*(-sin(ell_an)); % Initial X on ellipse
    y_E=ye+rx*cos(w*t(i) + start_an)*sin(ell_an) + ry*sin(w*t(i) +
start_an)*(cos(ell_an)); % Initial Y on ellipse
```

```
    X_C_O= [x_C,y_C]';
    X_L_C= TR_C2O*[L1,L2]';
    X_L_O= X_C_O+X_L_C; % Position of Look Ahead point in {O} frame= Xco +
Xlc
    x_L=X_L_O(1);
    y_L=X_L_O(2);

    X_Rt_C= TR_C2O*[-d,-b]';
    X_Rt_O= X_C_O+X_Rt_C; % Position of Look Ahead point in {O} frame= Xco
+ Xlc
    x_Rt=X_Rt_O(1);
    y_Rt=X_Rt_O(2);

    X_Lt_C= TR_C2O*[-d,b]';
    X_Lt_O= X_C_O+X_Lt_C; % Position of Look Ahead point in {O} frame= Xco
+ Xlc
    x_Lt=X_Lt_O(1);
    y_Lt=X_Lt_O(2);

    err1(i)= sqrt((x_E-x_L)^2 + (y_E-y_L)^2);

    plot([x_Rt,x_Lt],[y_Rt,y_Lt],'r');% Axle
    plot([(x_Rt+x_Lt)/2,x_C],[(y_Rt+y_Lt)/2,y_C],'r'); % Plot driver shaft
    plot(x_L,y_L,'bo'); % PLot Look ahead point
    plot(x_C,y_C,'mo'); % Plot CM

    pause(0.1);     %Stop execution for 0.01 to make animation visible
    %     frame= getframe(gcf);   %Step 2: Grab the frame
    %     aviobj = addframe(aviobj,frame); % Step 3: Add frame to avi
object
end
% aviobj = close(aviobj)  % Close the avi object
hold off
% axis equal;

figure(index*3-1);
plot(t,err1);
```