2009

Hrishi Shah
Sourish Chakravarty

KINEMATIC AND DYNAMIC CONTROL OF A TWO LINK MANIPULATOR

In this homework, we explore the development of various control schemes in the context of a Two Link Manipulator and compare their performance via their simulations while tracing a rotated ellipse.

Contents

A) KINEMATIC ANALYSIS

The first part of the homework was kinematic level analysis of the afore-stated system.

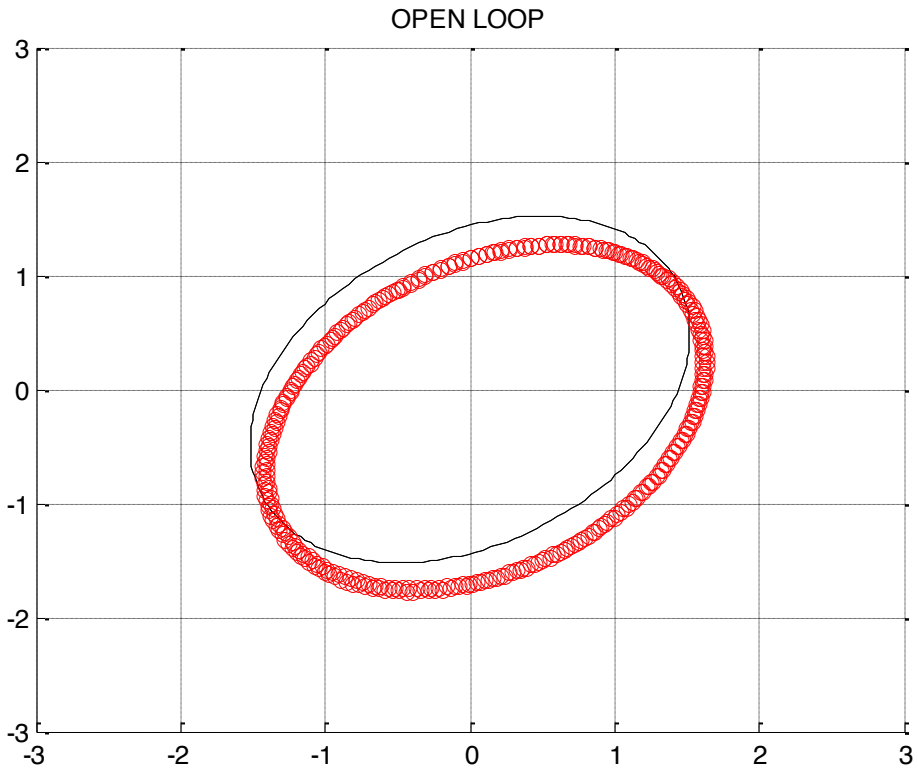
The position level equations were written and differentiated to get the velocity level equations.

1) OPEN LOOP

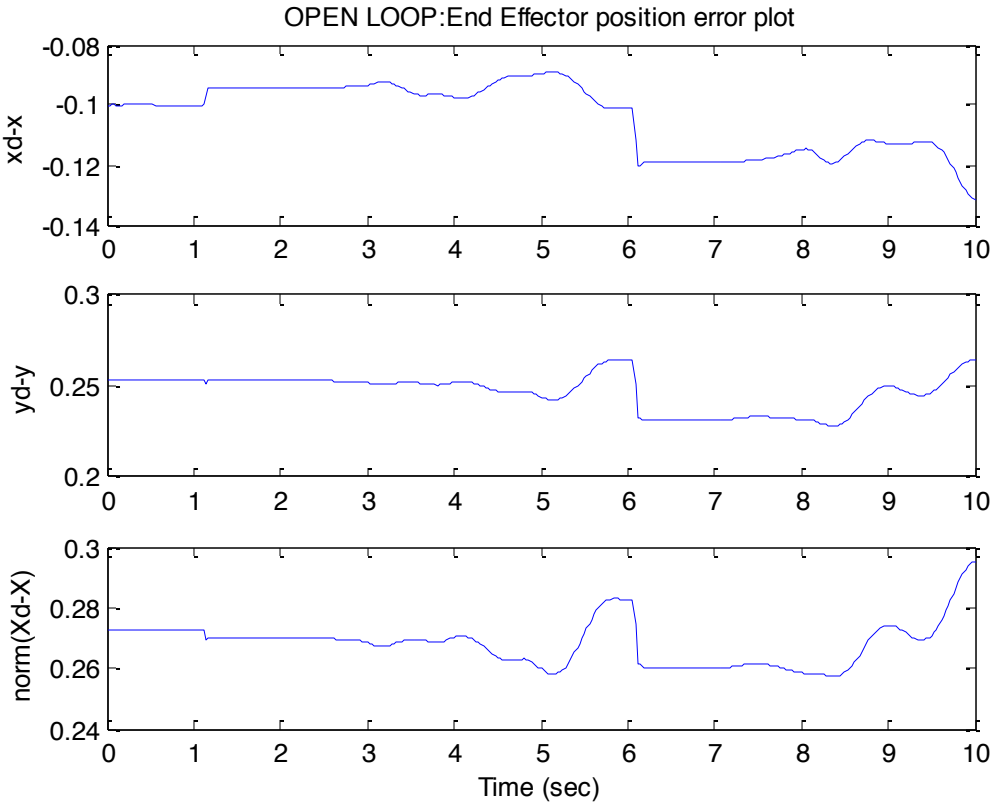
The governing equation is:

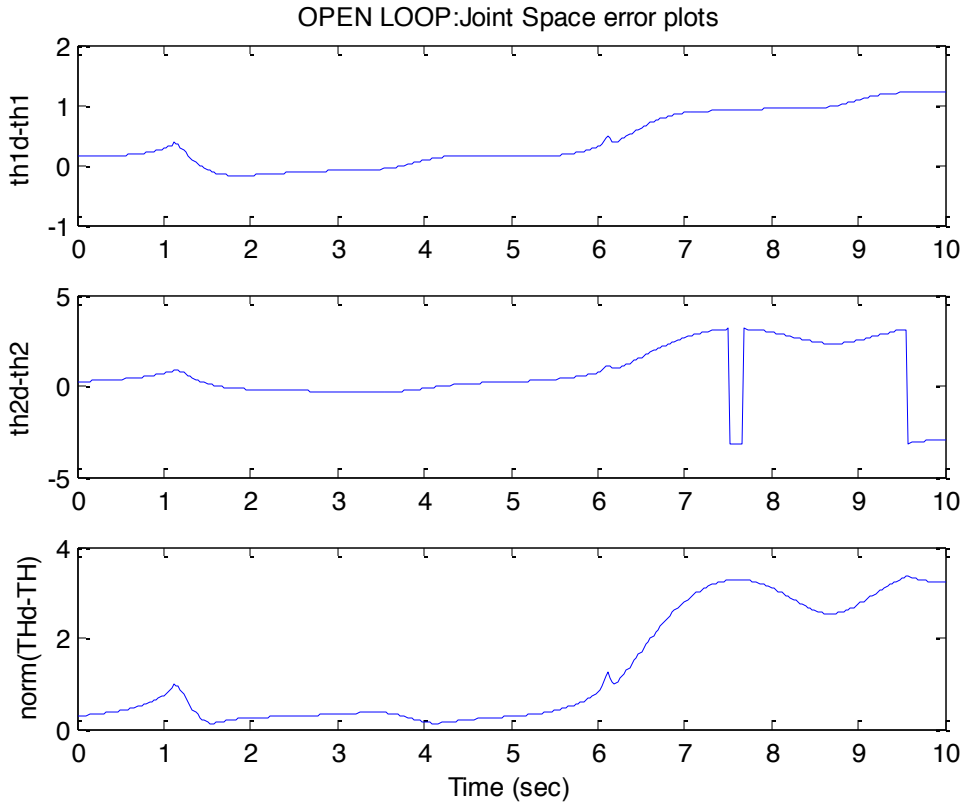
$$\dot{\theta} = \dot{\theta}^d = J(\theta)^{-1} \cdot \dot{X}$$

There is no control involved. Hence, the initial disturbance in the system prevails throughout the simulation. Since there are no 'actual position' dependent terms in the simulation, the simulation is stable and consistent despite the disturbance. Hence, the shape is simply a shifted version of the expected shape.



The errors are as plotted below. Since the x and y values are non-linear, the differences between actual and desired x and y values are slightly non-linear.

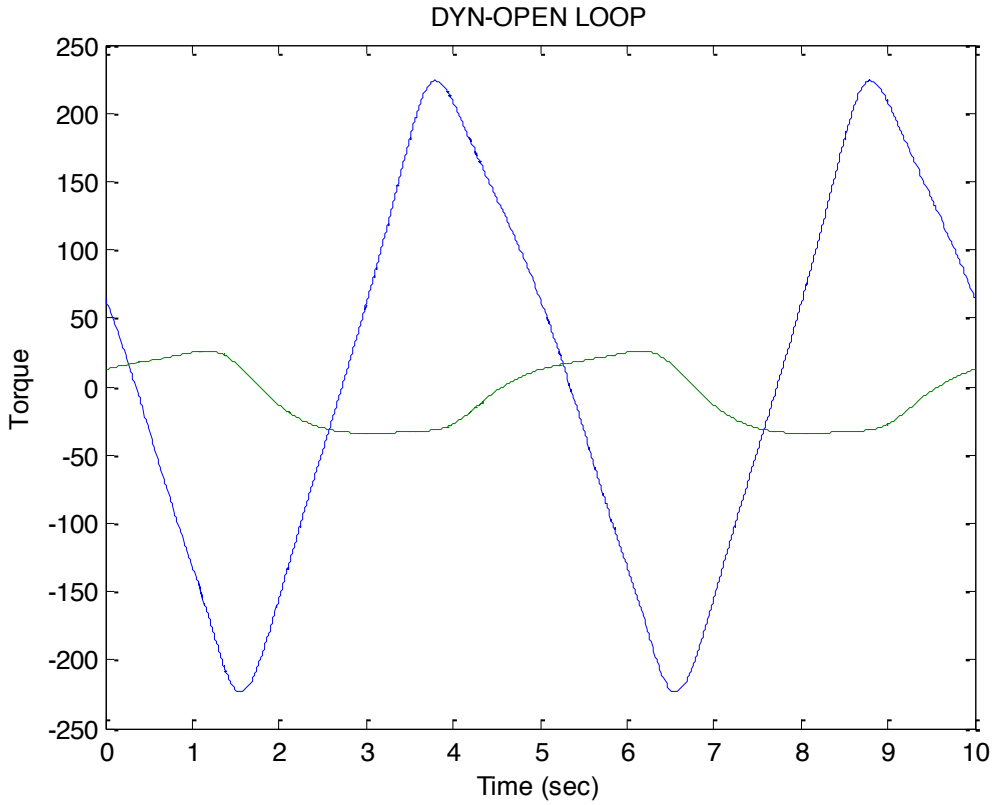

OPEN LOOP:End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)


OPEN LOOP:Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

Now, the code is changed to produce $\ddot{\theta}$ also. The ode45 function is still running on only $\dot{\theta}$ and θ. This allows for calculation of torque profiles required by the two motors to get the required motion. The governing equation is:

$$\tau = M(\theta)\ddot{\theta} + V\left(\theta, \dot{\theta}\right) + G(\theta)$$

The torque profiles are plotted below. The larger one is the base motor torque.



2) JOINT SPACE CONTROL

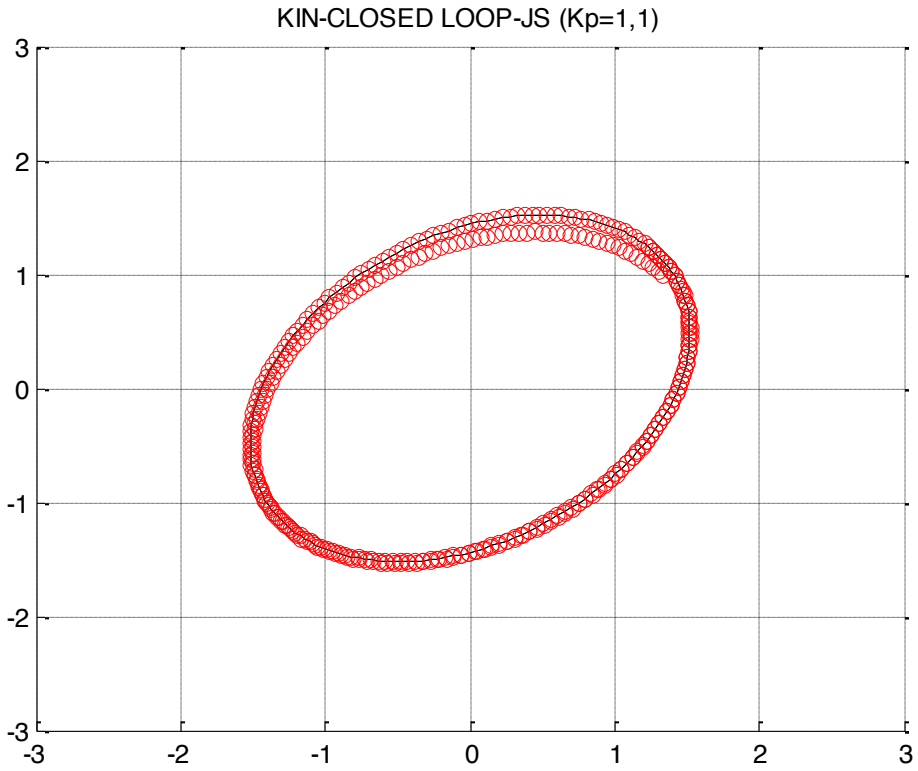
Now, a control term relating to the difference in the desired and actual values of joint angles in introduced into the equation, which now becomes:

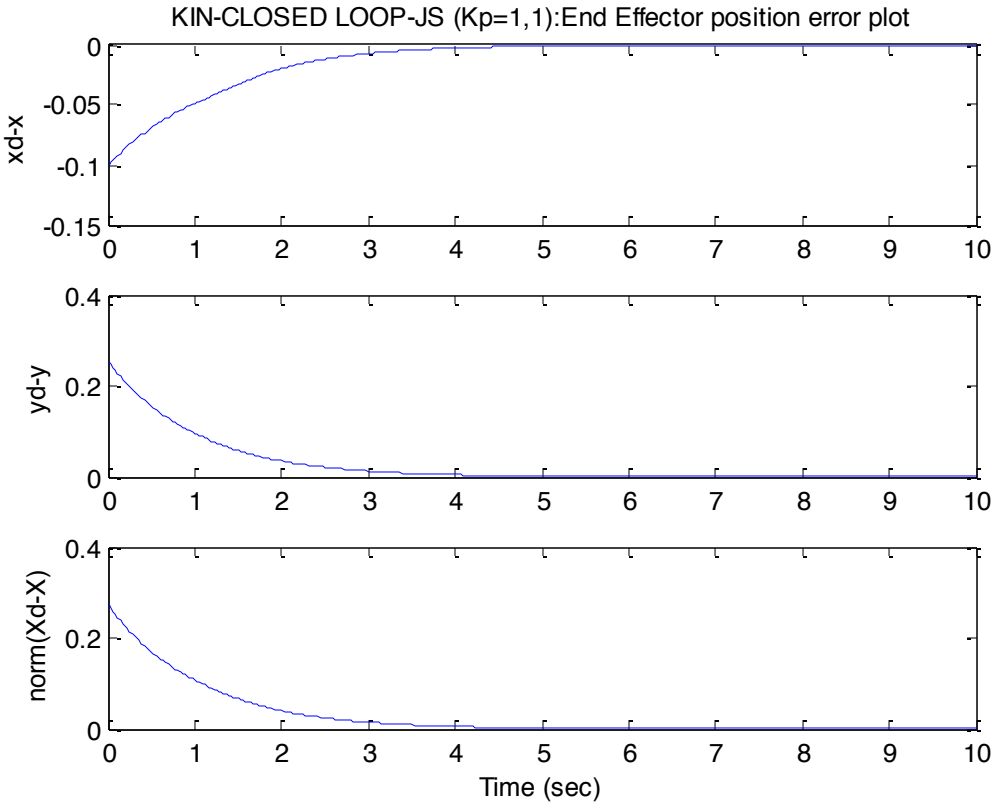
$$\dot{\theta} = J(\theta)^{-1} \cdot \dot{X} + K_p(\theta_d - \theta)$$

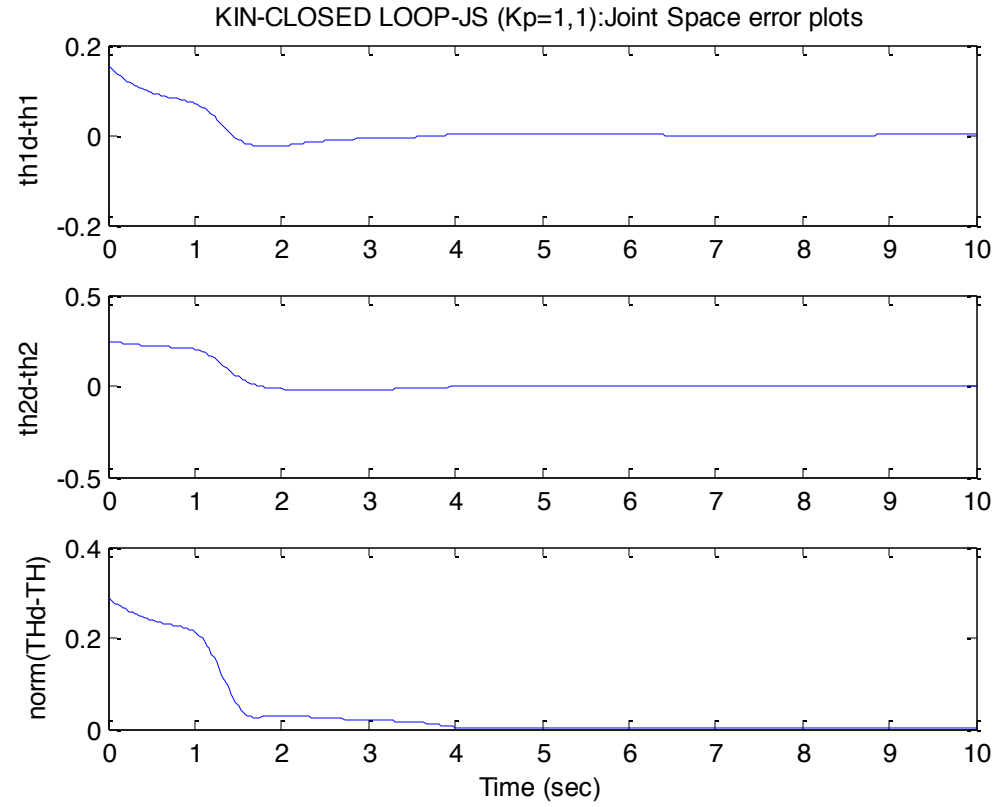
This introduces two parameters K_{p1} and K_{p2} (for two joint angles), into the equation. Various combinations were tried out.

a) $K_{p1}=K_{p2}=1$

As can be seen, the convergence is slow. This is confirmed by the error plots that follow.

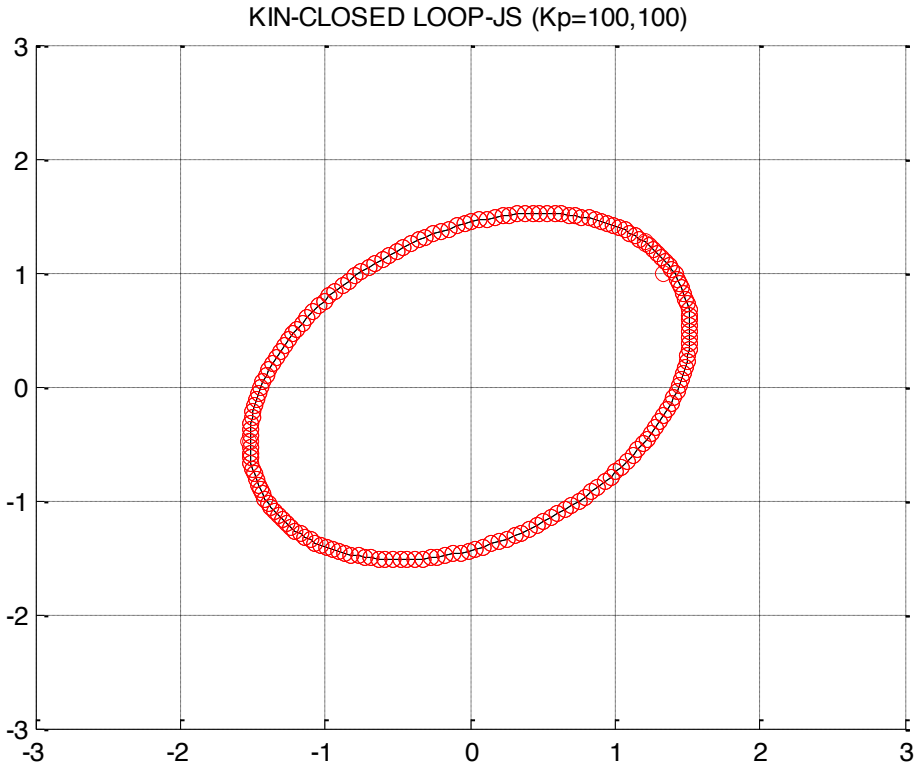


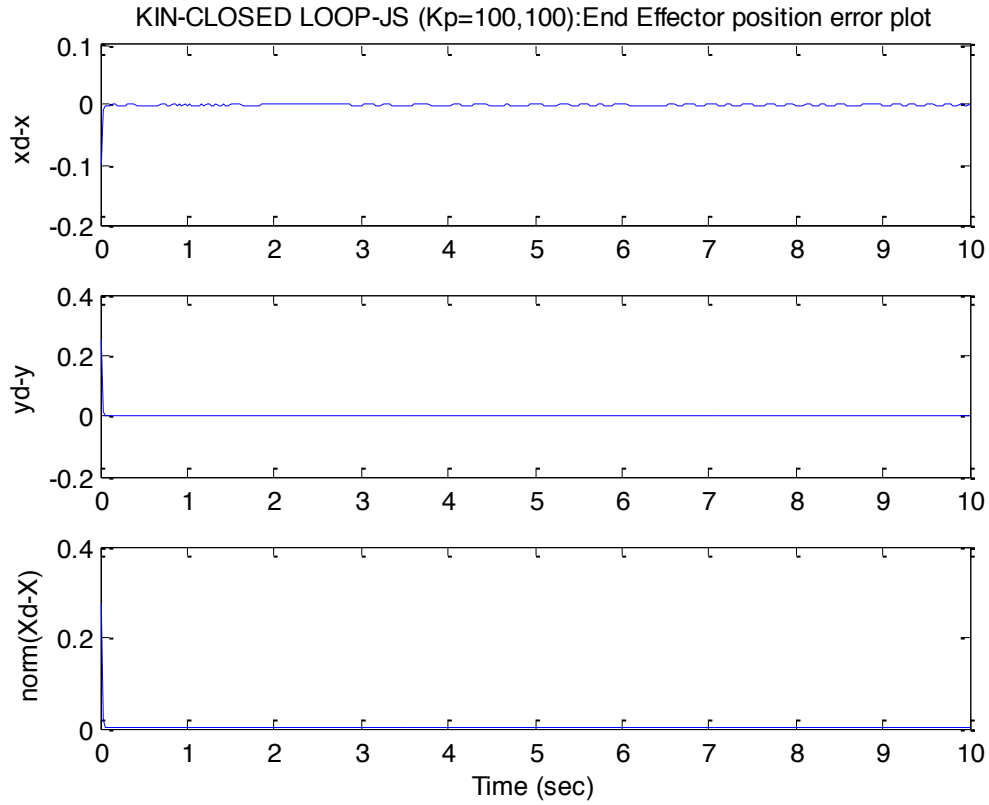

KIN-CLOSED LOOP-JS (Kp=1,1):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

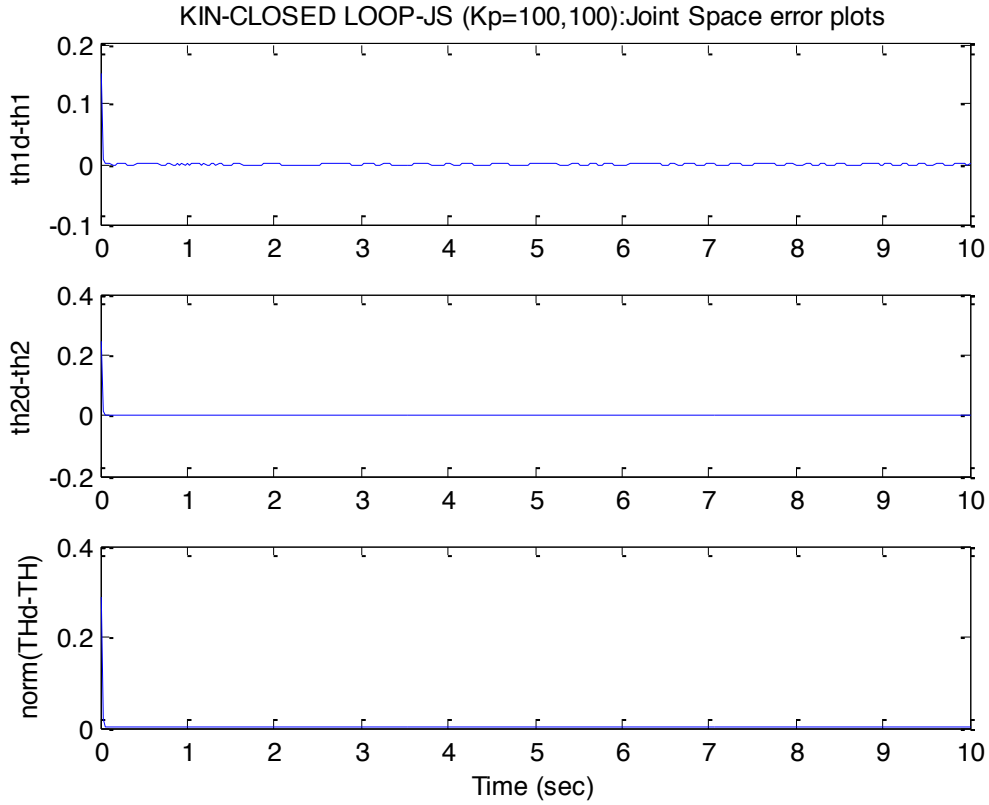

KIN-CLOSED LOOP-JS (Kp=1,1):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

b) $K_{p1}=K_{p2}=1$

In this case, both values were simultaneously increased. Thus, faster convergence is observed in both joint angles and hence a much better performance than the previous case. This is again confirmed by the error plots that follow.

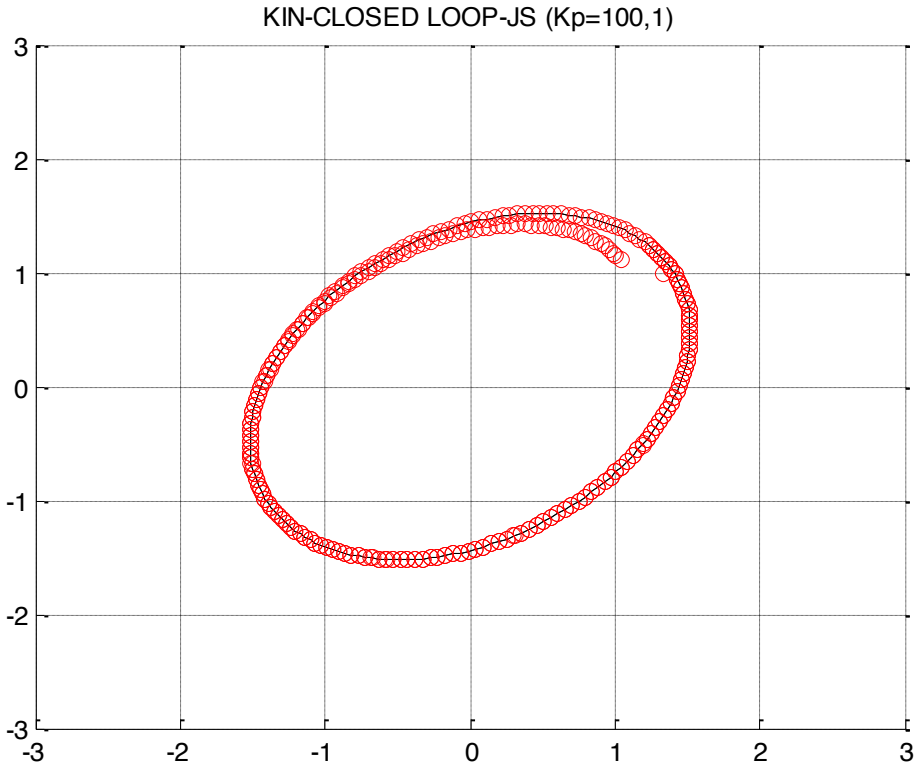


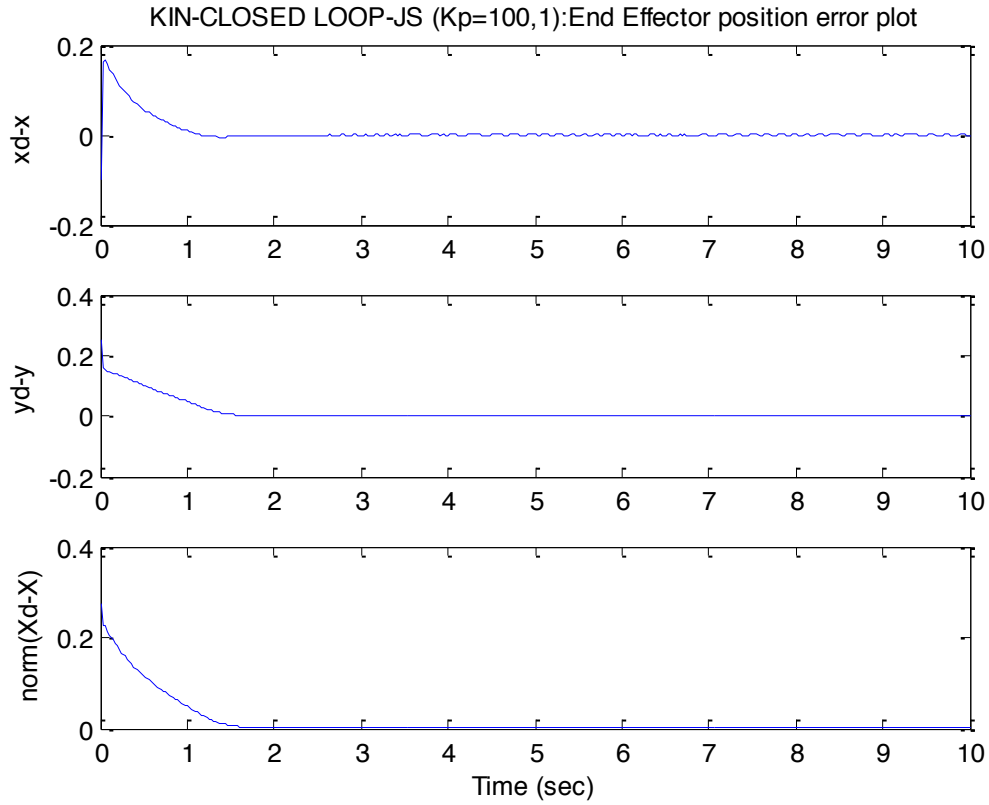

KIN-CLOSED LOOP-JS (Kp=100,100):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

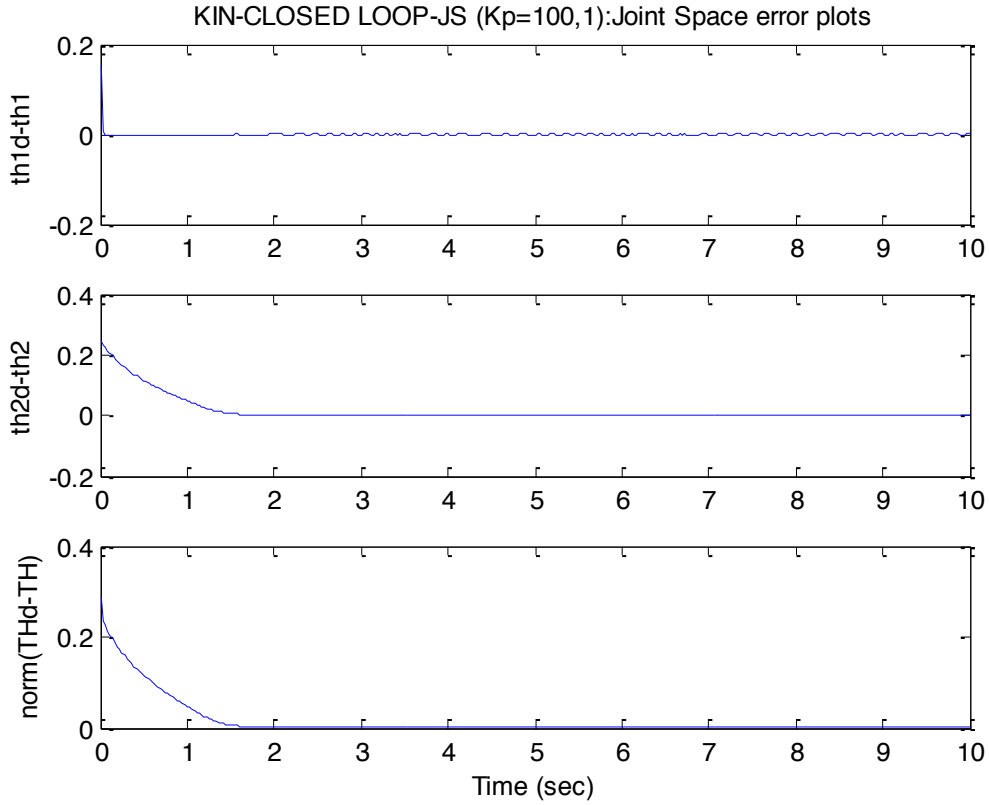

KIN-CLOSED LOOP-JS (Kp=100,100):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

c) $K_{p1}=100, K_{p2}=1$

In this case, only the first joint angle is given a high weight in control. Hence, only that angle converges faster. The other angle has a slower convergence. Hence the task space error is also intermediate (between case 1 and case 2).




KIN-CLOSED LOOP-JS (Kp=100,1):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)


KIN-CLOSED LOOP-JS (Kp=100,1):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

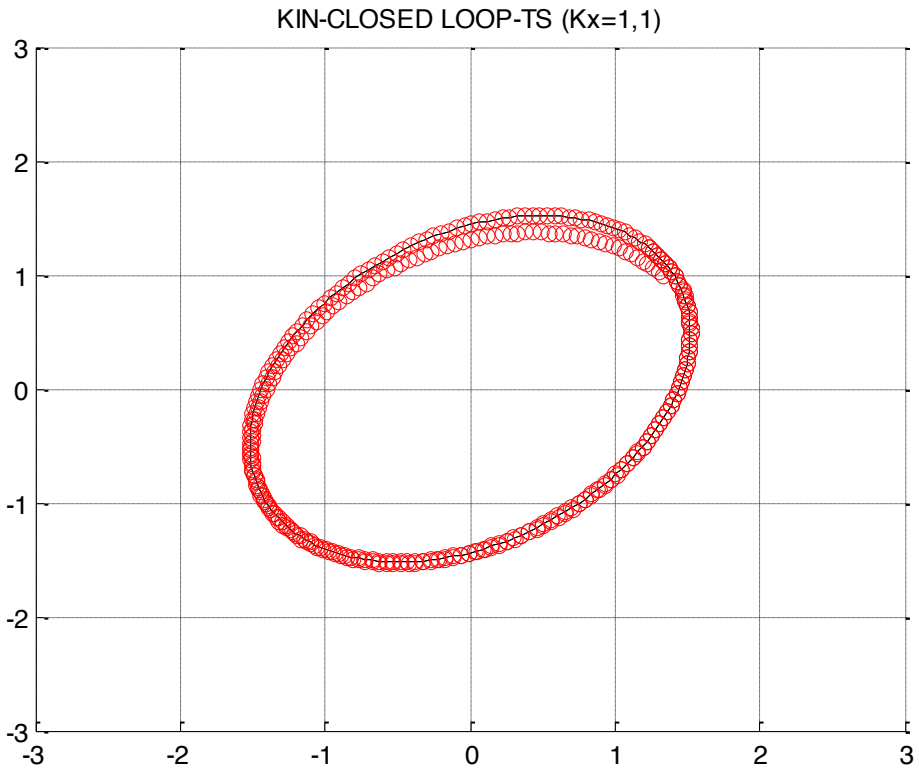
3) TASK SPACE CONTROL

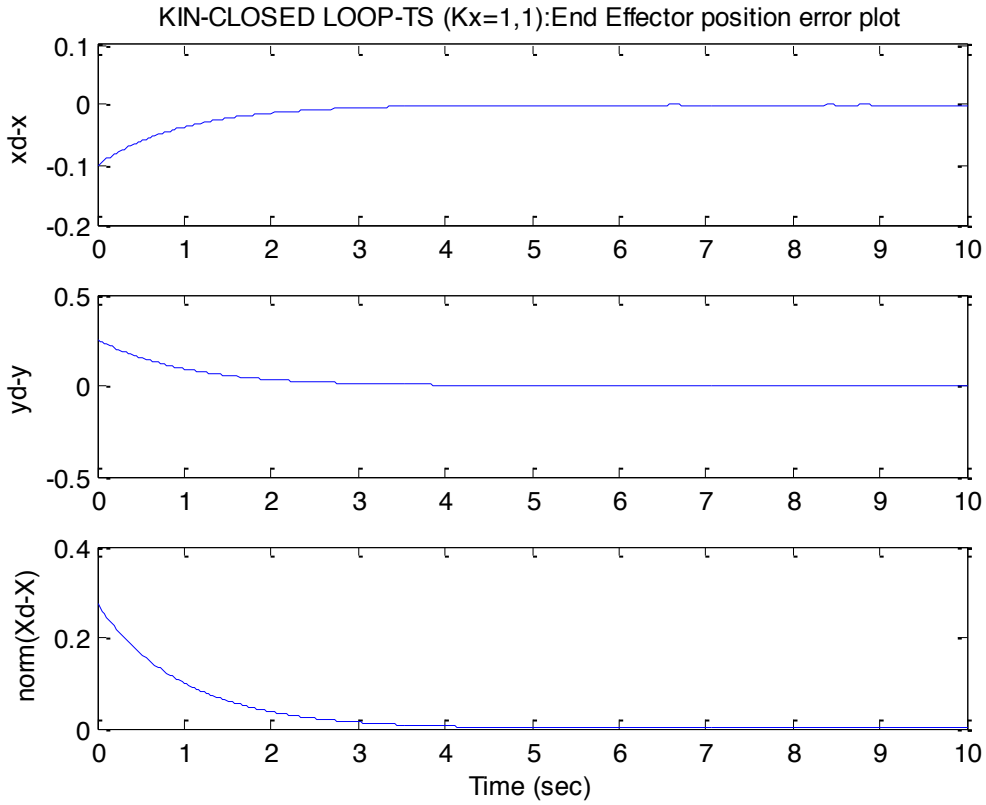
In this case, the differences between the desired and actual task space values are used as error signals and converted to joint space by the inverse of the Jacobian. Similar to Joint Space, we have K_{x1} and K_{x2} as parameters that are varied and results explored. The governing equation is:

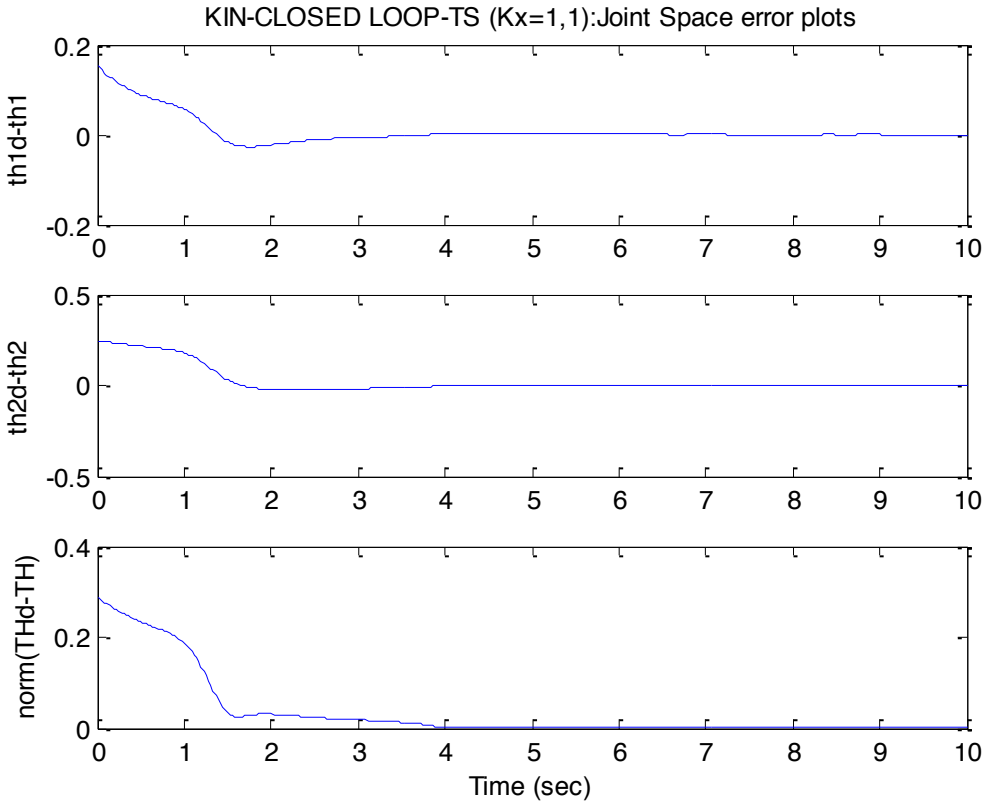
$$\dot{\theta} = J(\theta)^{-1} \cdot \{\dot{X} + K_x(X_d - X)\}$$

a) $K_{x1}=K_{x2}=1$

As seen in the similar joint space scenario, convergence is slow and this is reflected in the error plots.

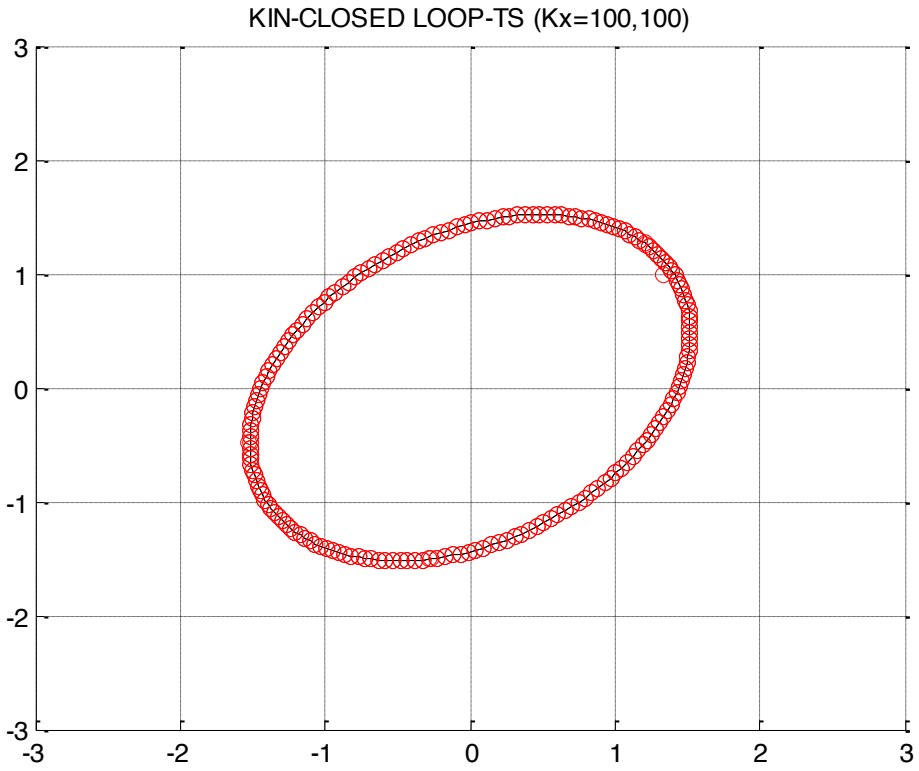


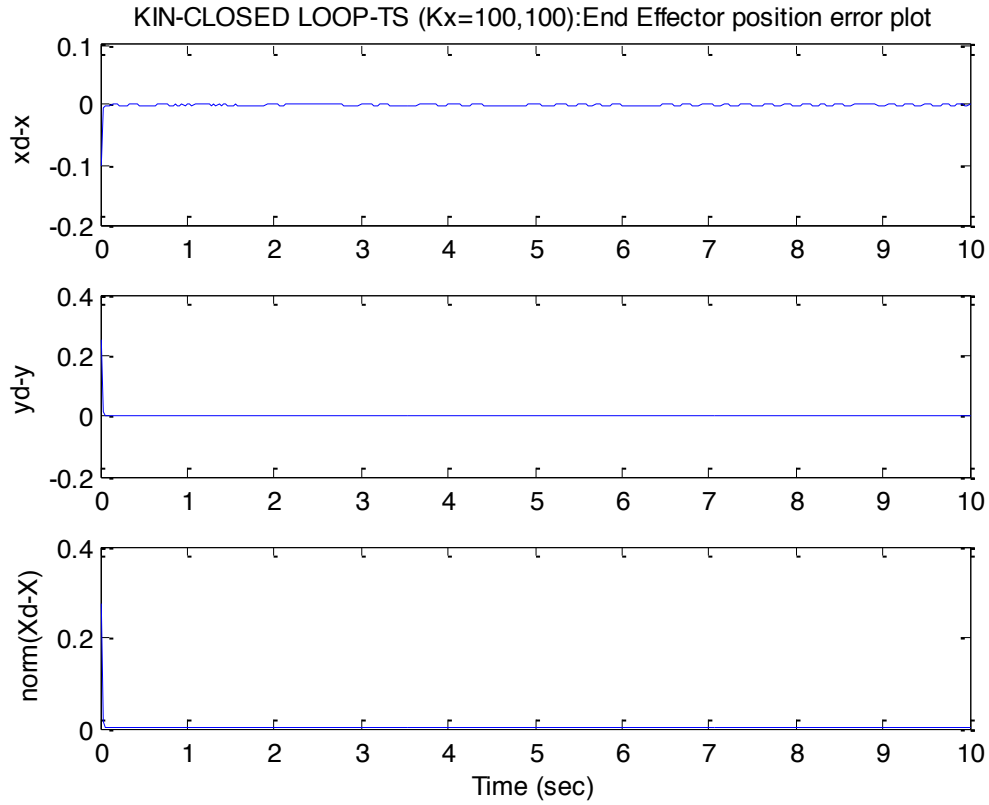

KIN-CLOSED LOOP-TS (Kx=1,1):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

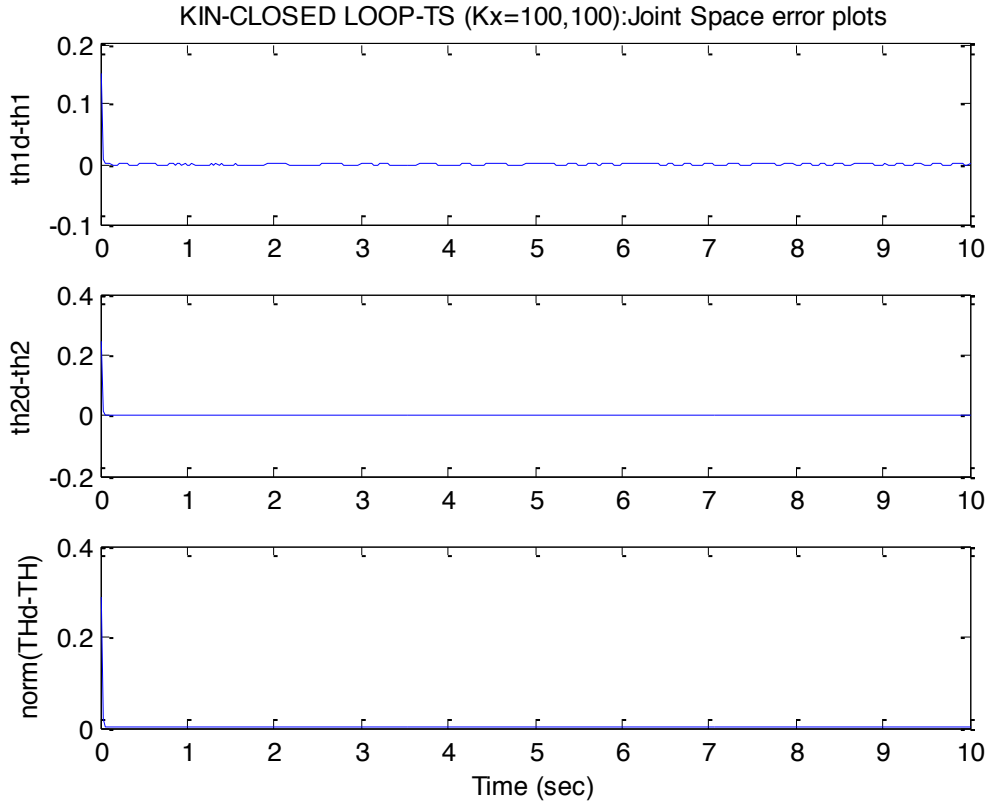

KIN-CLOSED LOOP-TS (Kx=1,1):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

b) $K_{x1}=K_{x2}=100$

Both the K values are increased simultaneously and hence a faster convergence is observed. One more thing to notice is that the task space errors are less in this case than in the corresponding joint space control scenario. This is expected as joint space errors get multiplied by the link length to enter the task space and hence relatively smaller joint space errors produce larger task space errors.

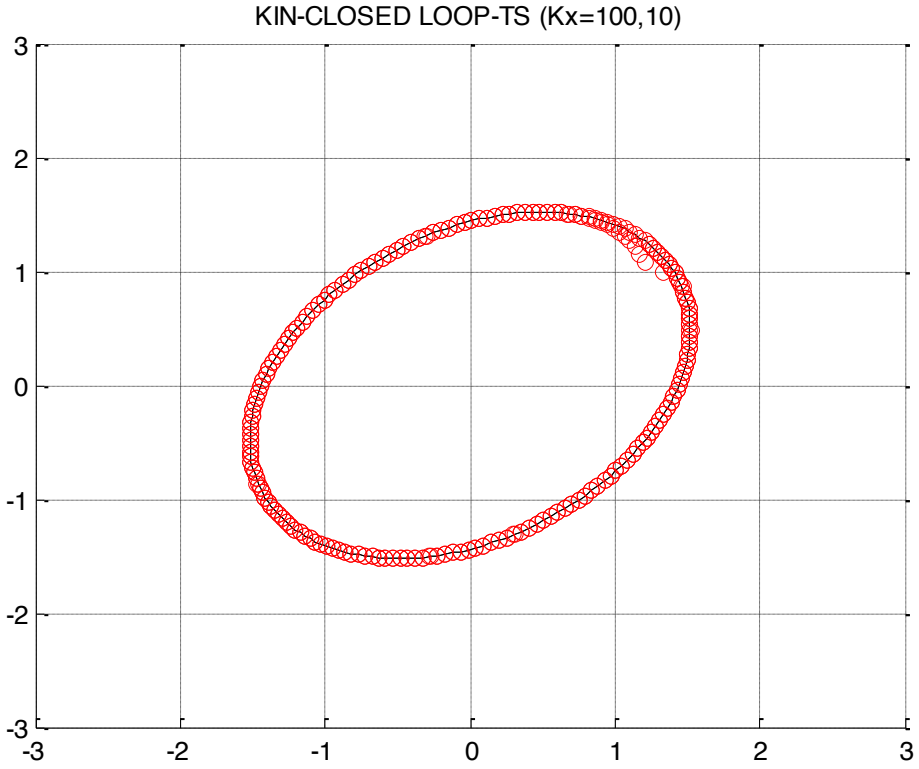


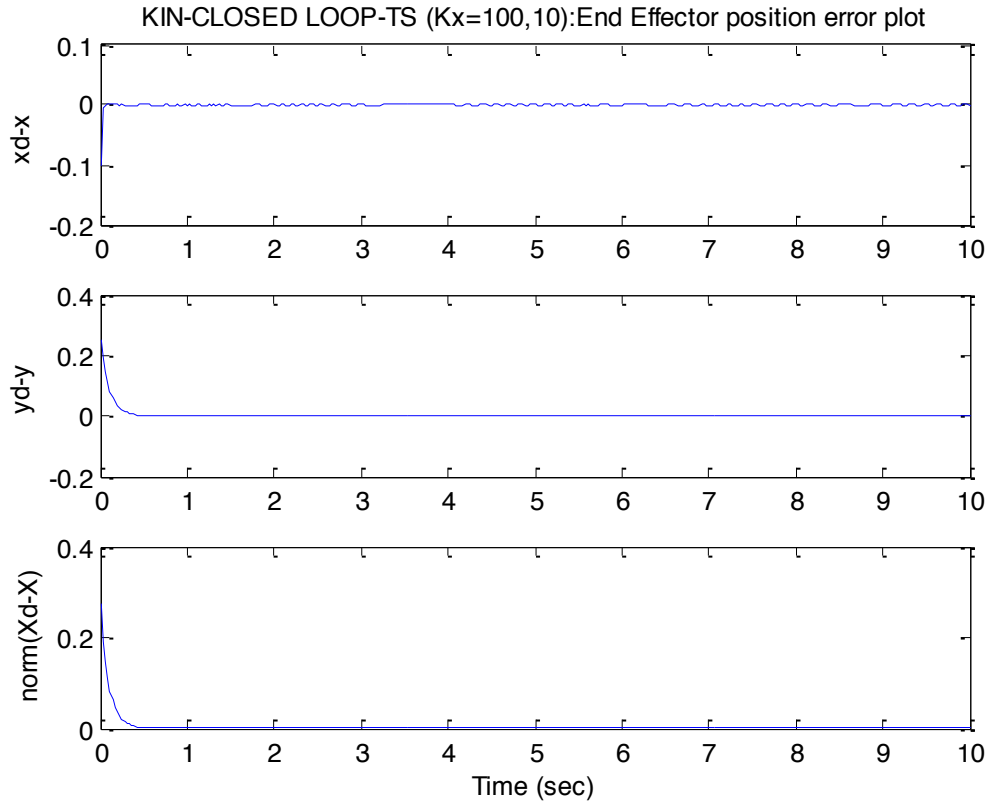

KIN-CLOSED LOOP-TS (Kx=100,100):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

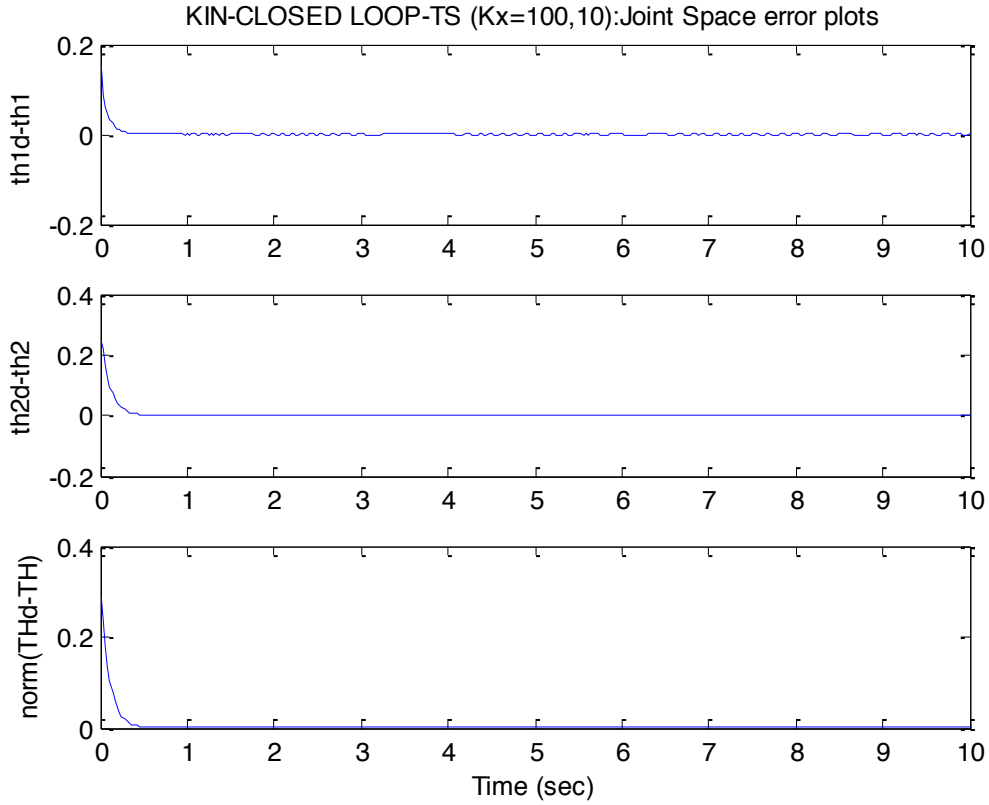

KIN-CLOSED LOOP-TS (Kx=100,100):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

c) $K_{x1}=100, K_{x2}=10$

In this case only the x-direction errors have a high weight in control and hence they converge faster than corresponding y-direction errors. It can be observed clearly in the error plots.




KIN-CLOSED LOOP-TS (Kx=100,10):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)


KIN-CLOSED LOOP-TS (Kx=100,10):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

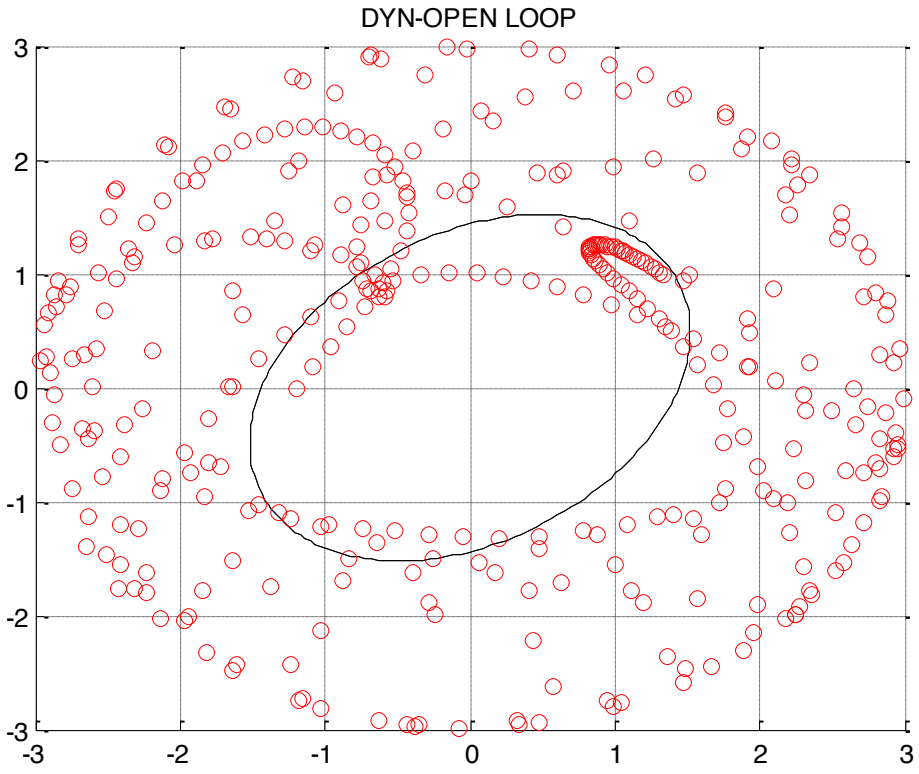
B) DYNAMIC ANALYSIS

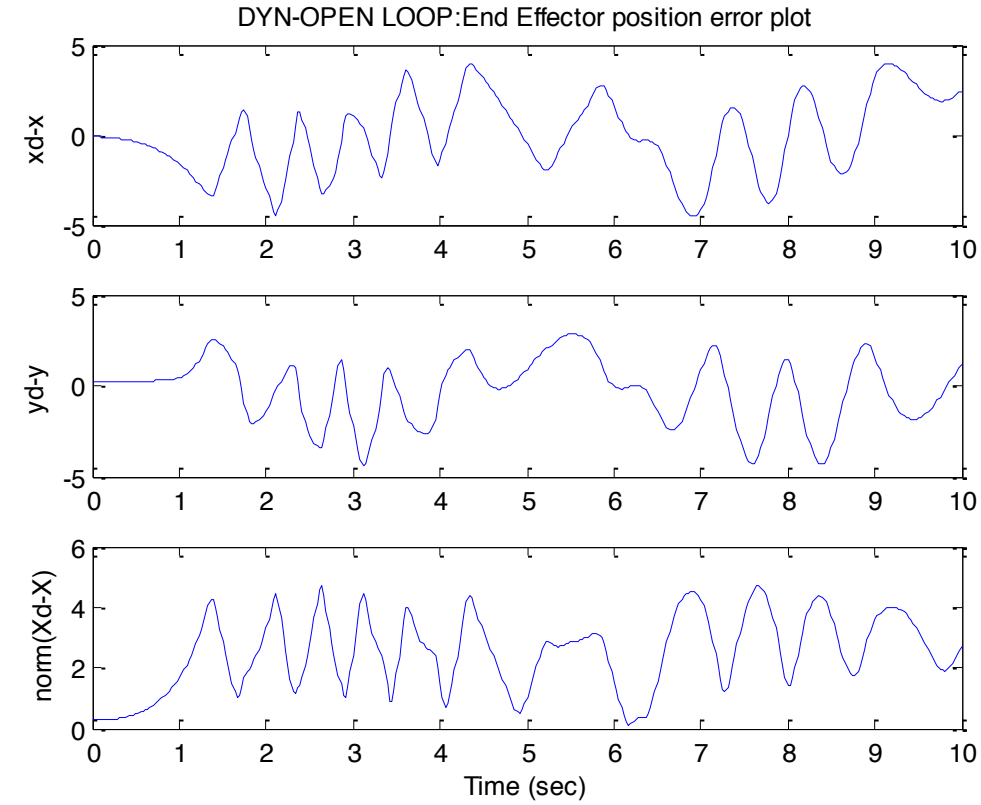
In this scenario, we have the inertia terms and torques entering the system analysis. The governing equation (as discussed before) is

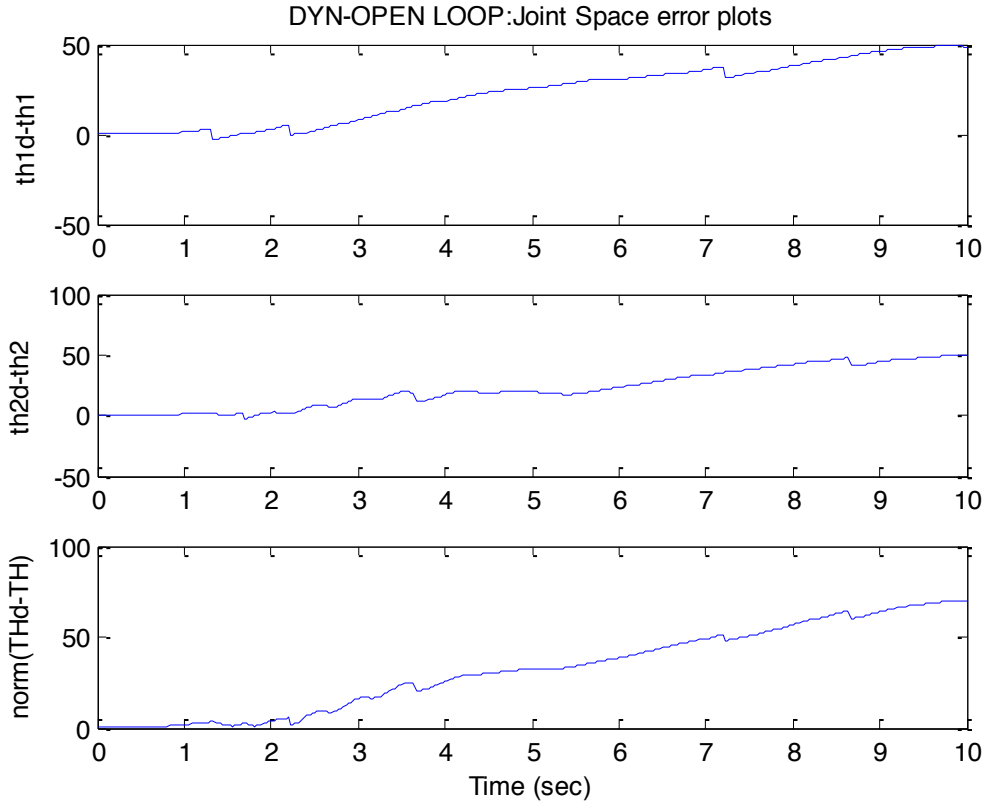
$$\tau = M(\theta)\ddot{\theta} + V\left(\theta, \dot{\theta}\right) + G(\theta)$$

1) OPEN LOOP

In this case, we just drive the system with the torques derived from the joint space derivatives derived from the desired joint space variables and their derivatives directly. In contrast to Kinematic domain, the initial disturbance causes the system to become unstable almost instantly as there are gravity terms in the equation that are dependent on 'actual position'.

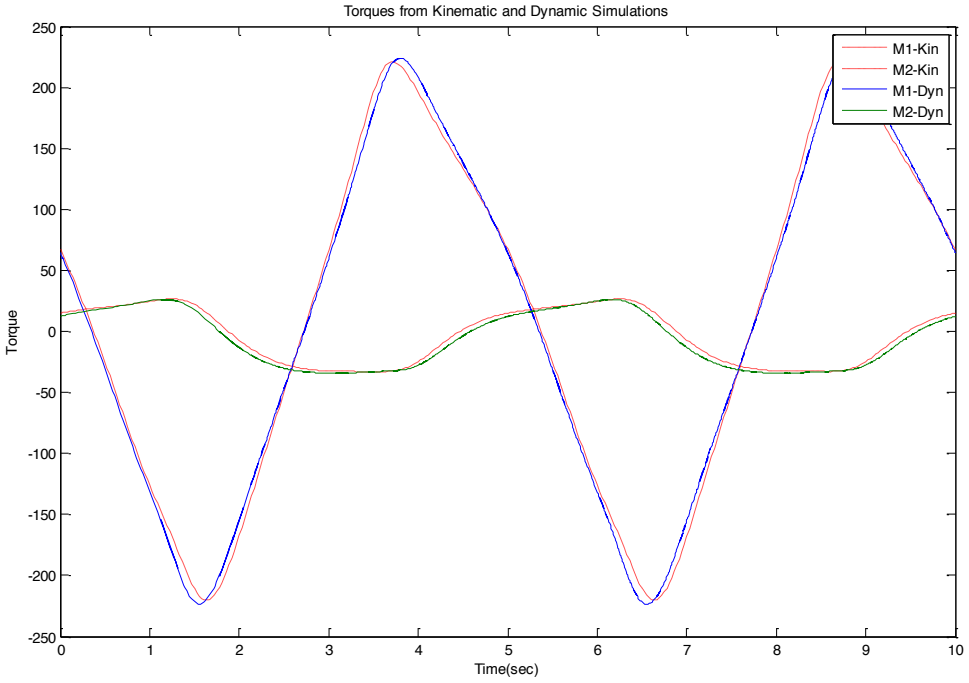



DYN-OPEN LOOP:End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

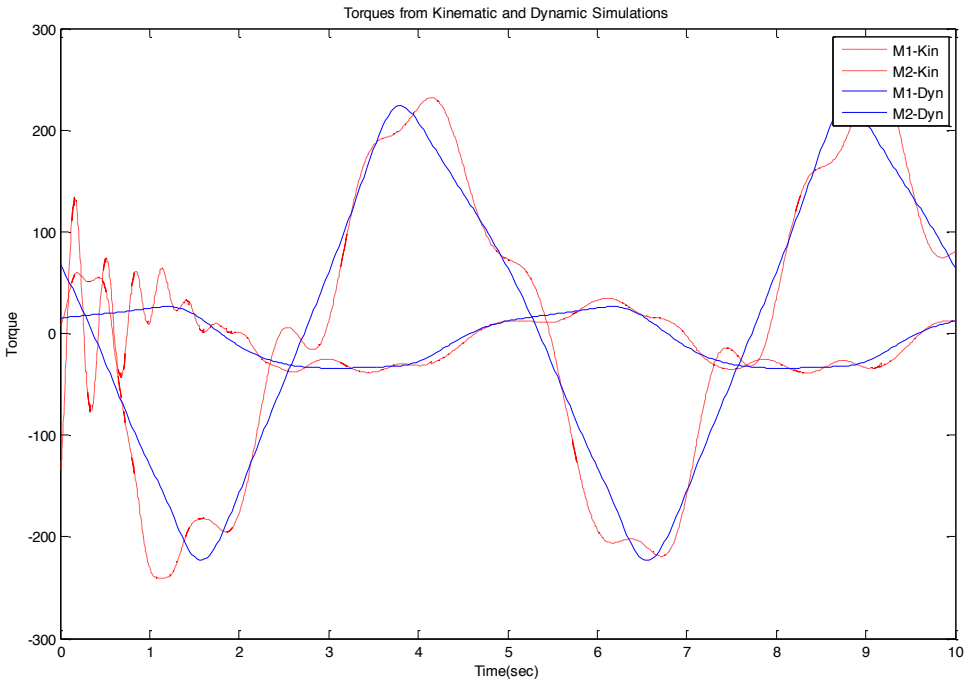

DYN-OPEN LOOP:Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

Special Topic: COMPARISON OF KINEMATIC AND DYNAMIC TORQUES

As a comparison of Kinematic and Dynamic Analyses, we trace the torques computed in both the cases and study their variation.



We also tried to compare using Closed Loop Simulation. Kin-JS_CL and Dyn-B1 are compared. The torques required in the Dynamic scenario are better conditioned but more in magnitude than their Kinematic counterparts. Sum (Kinematic Torques) = 1.0e+006 *[7.0538 0.2010]. Sum (Dynamic Torques) = 1.0e+007 *[3.0489 0.2006]



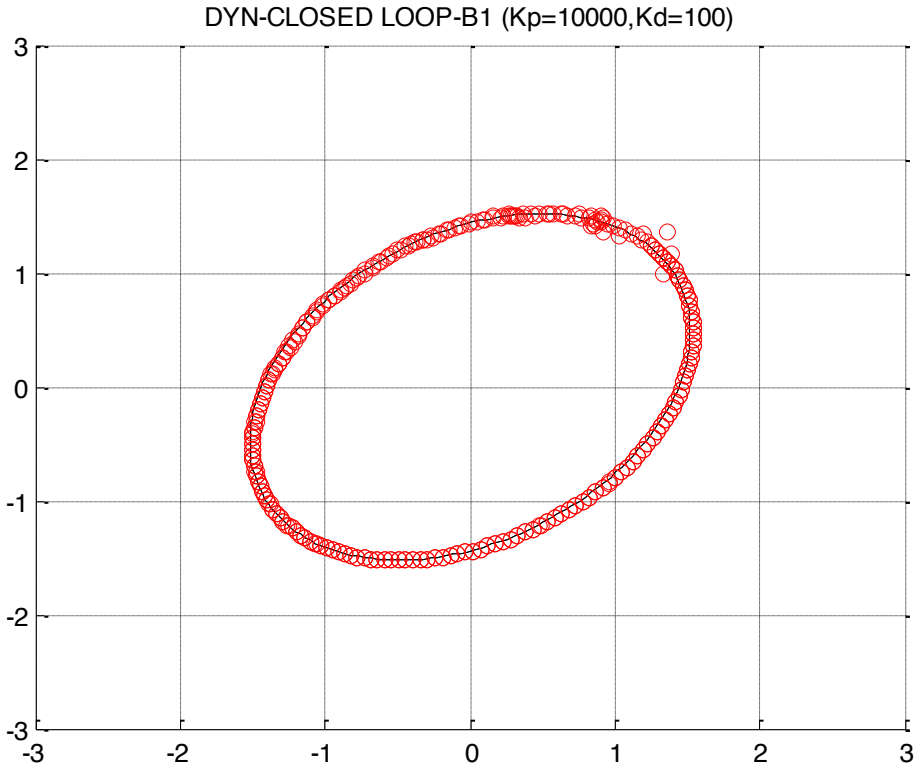
2) B1-DIRECT ERROR COMPENSATION

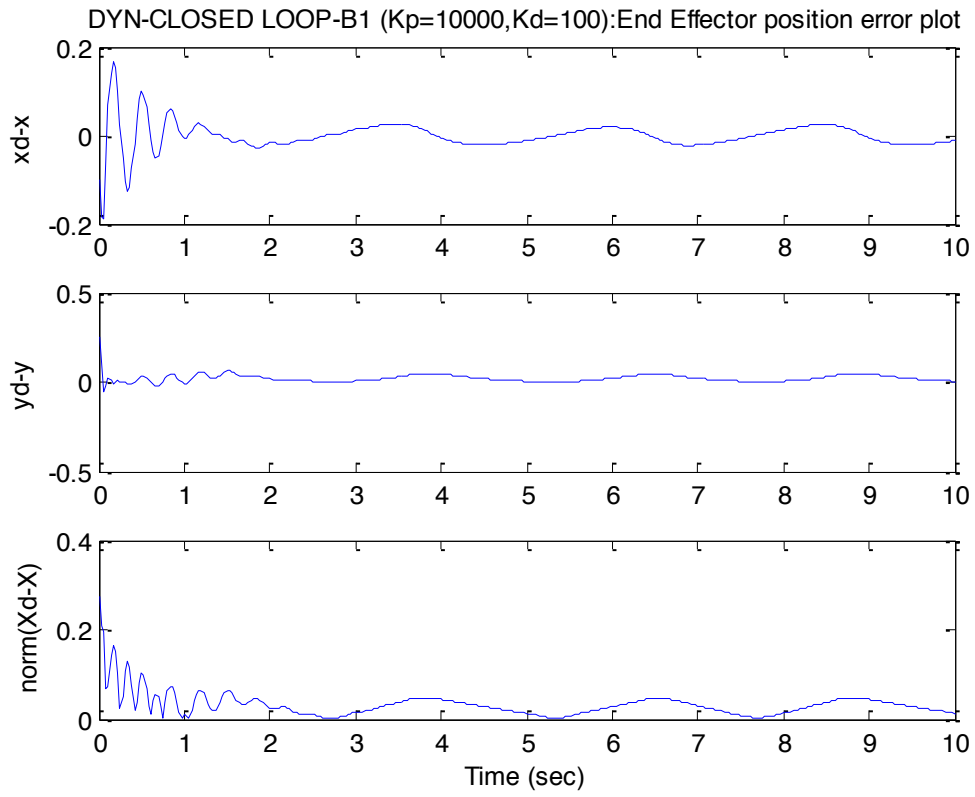
In this case, we provide corrective torques proportional to the errors in $\theta$and $\dot{\theta}$. The governing equation becomes

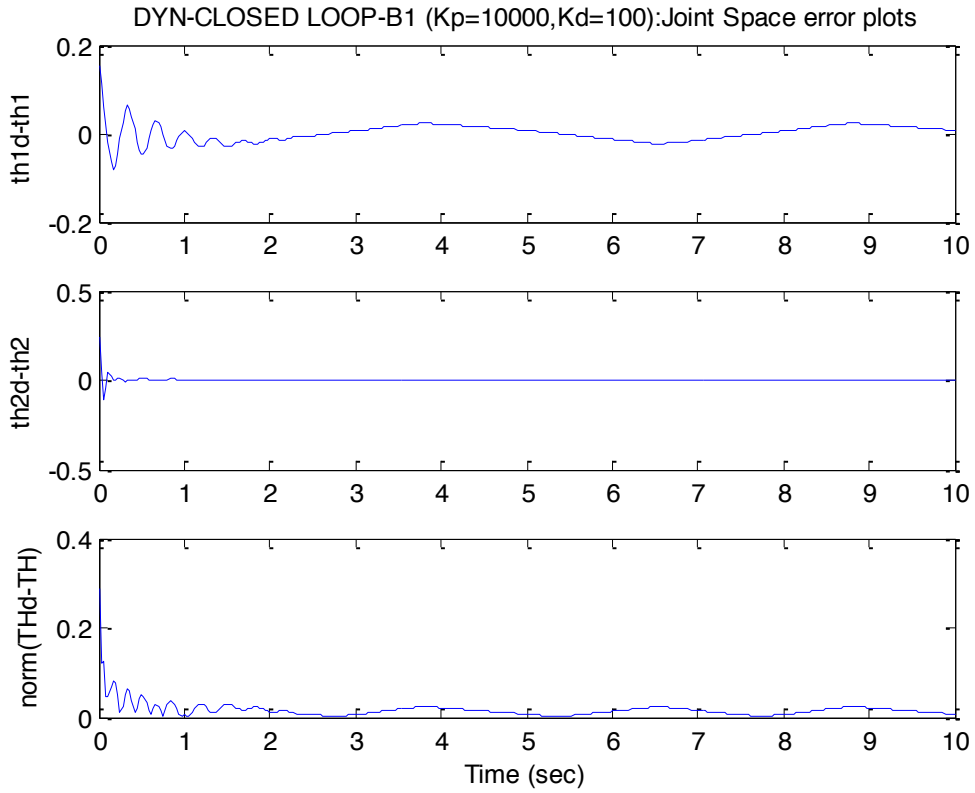
$$\tau = K_p\left(\theta^d - \theta\right) + K_d(\dot{\theta}^d - \dot{\theta})$$

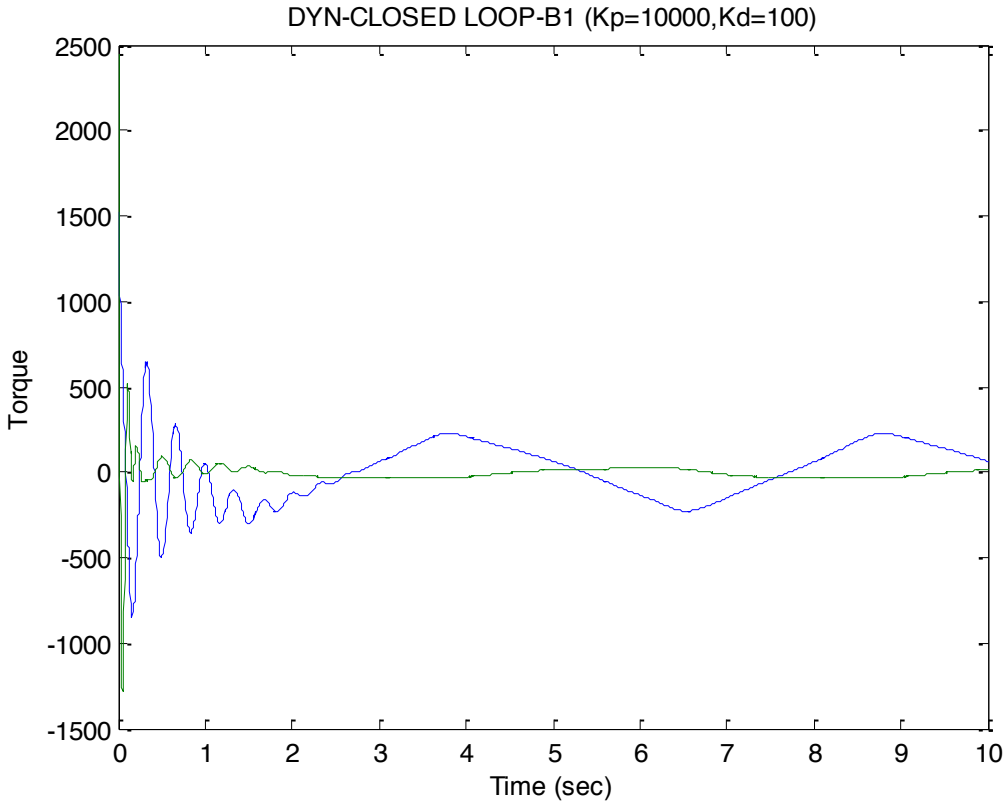
The K_p and K_d values are varied and system response studied.

a) K_p=10000(HIGH), K_d=100(HIGH)



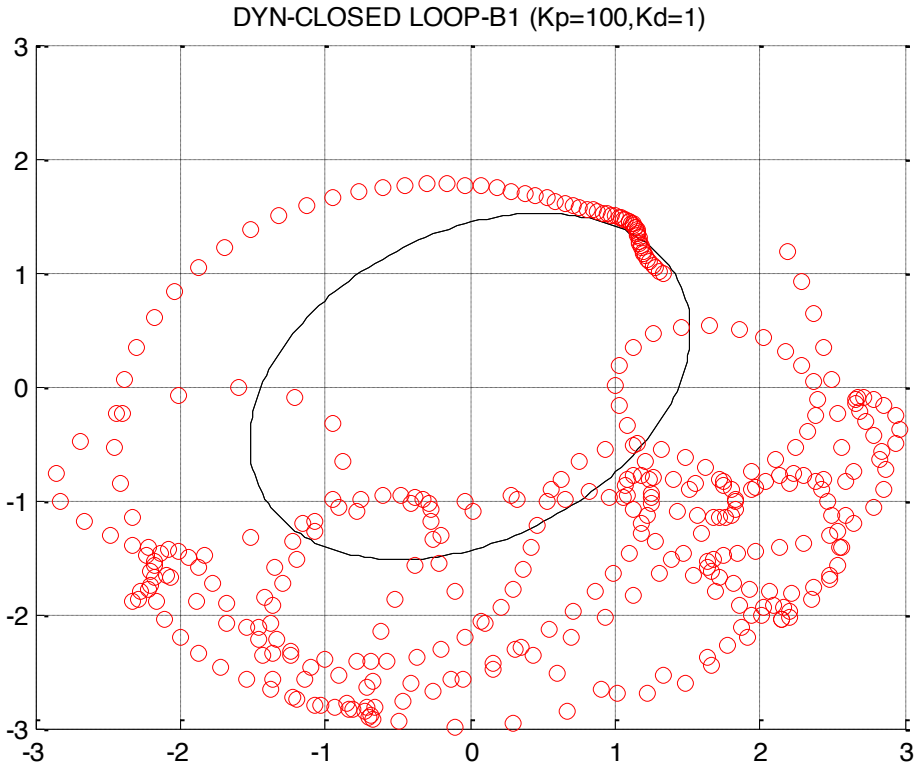

DYN-CLOSED LOOP-B1 (Kp=10000,Kd=100):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

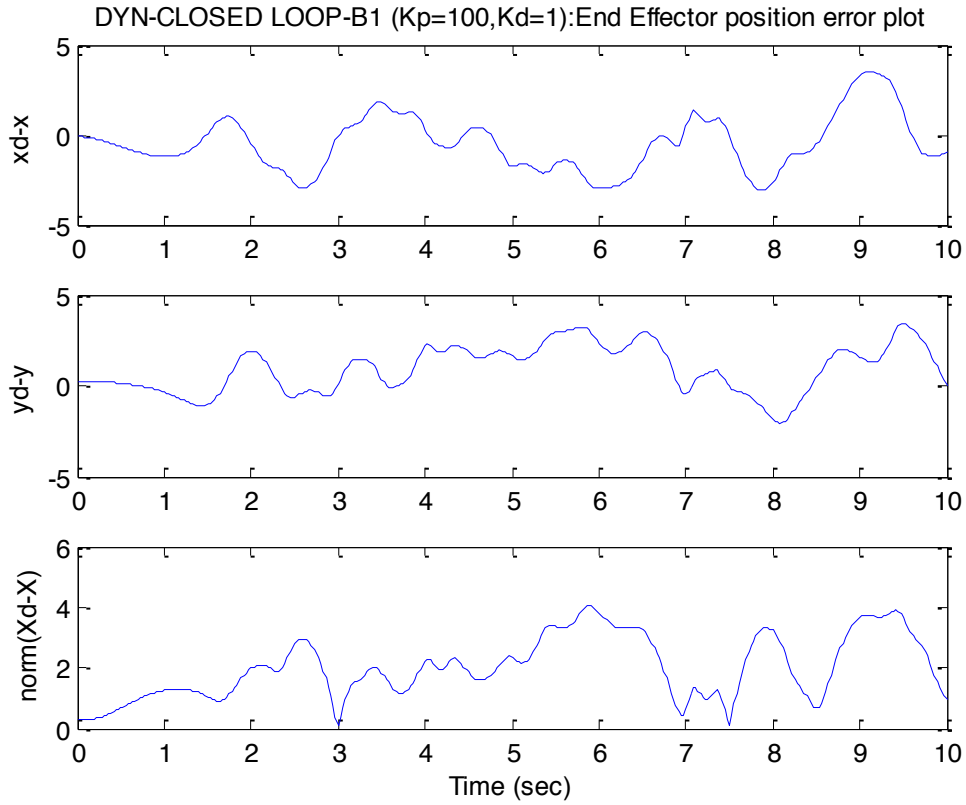

DYN-CLOSED LOOP-B1 (Kp=10000,Kd=100):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

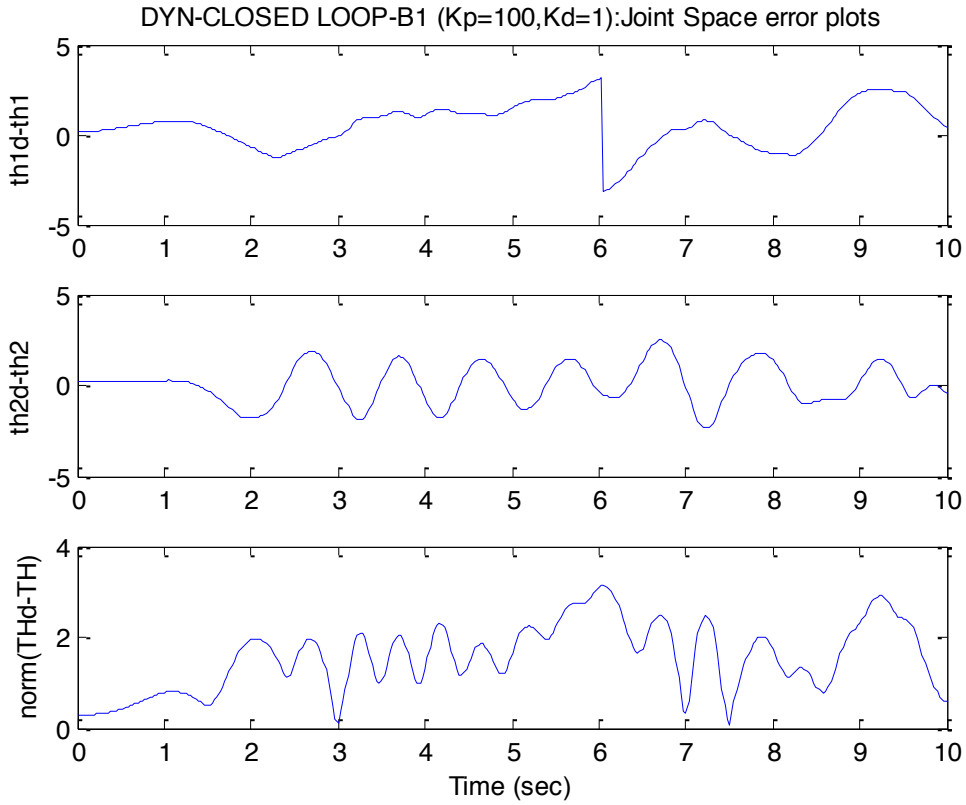

DYN-CLOSED LOOP-B1 (Kp=10000,Kd=100)
Torque
Time (sec)
2500
2000
1500
1000
500
0
-500
-1000
-1500
0
1
2
3
4
5
6
7
8
9
10

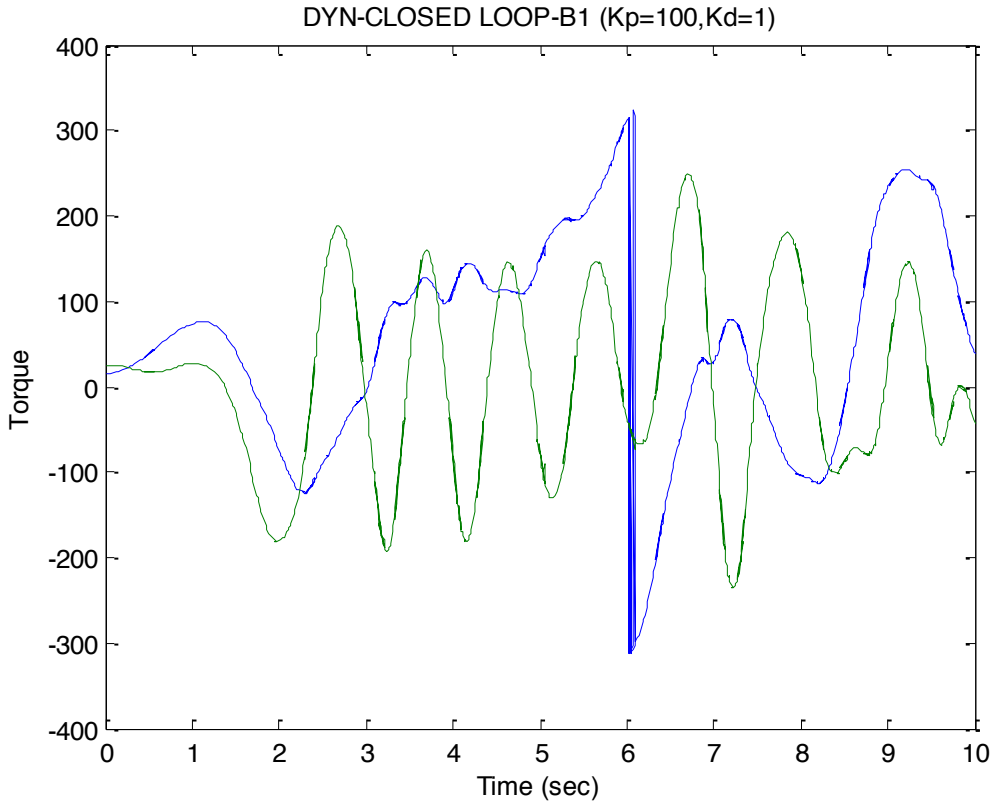
b) K_p=100(LOW), K_d=1(LOW)

In this case, we have low values for both K_p and K_d. We can see that the system has diverged at a point slightly after the point it diverged in open loop dynamic system. In that period, the control torque that is provided by these meager values of K_p and K_d is enough to combat the error in terms due to misjudged 'actual position' (i.e. initial disturbance).



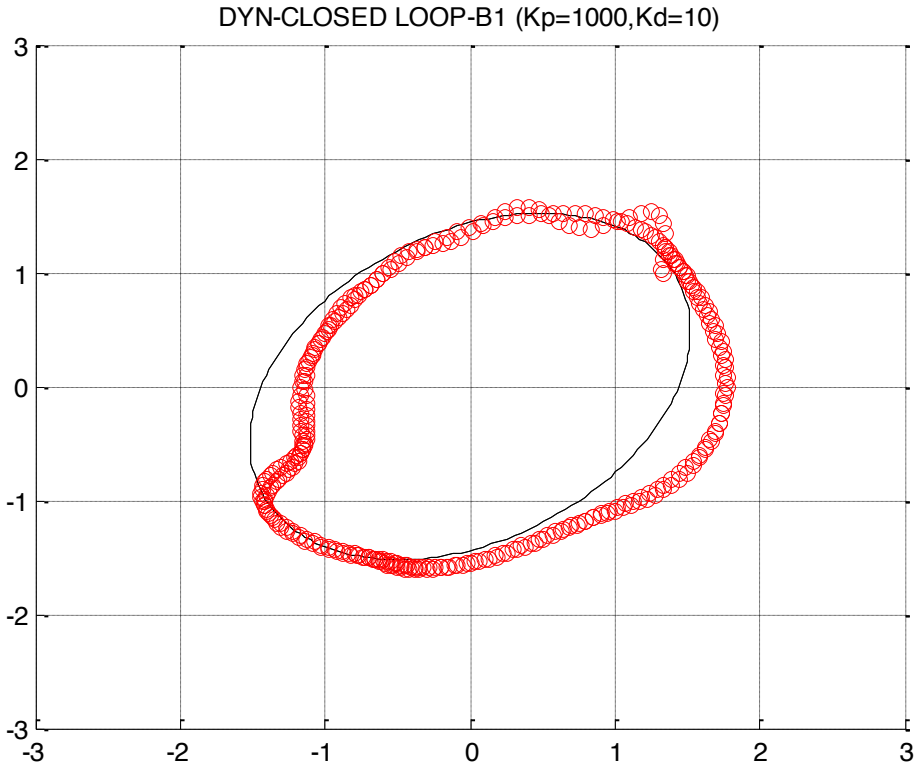

DYN-CLOSED LOOP-B1 (Kp=100,Kd=1):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

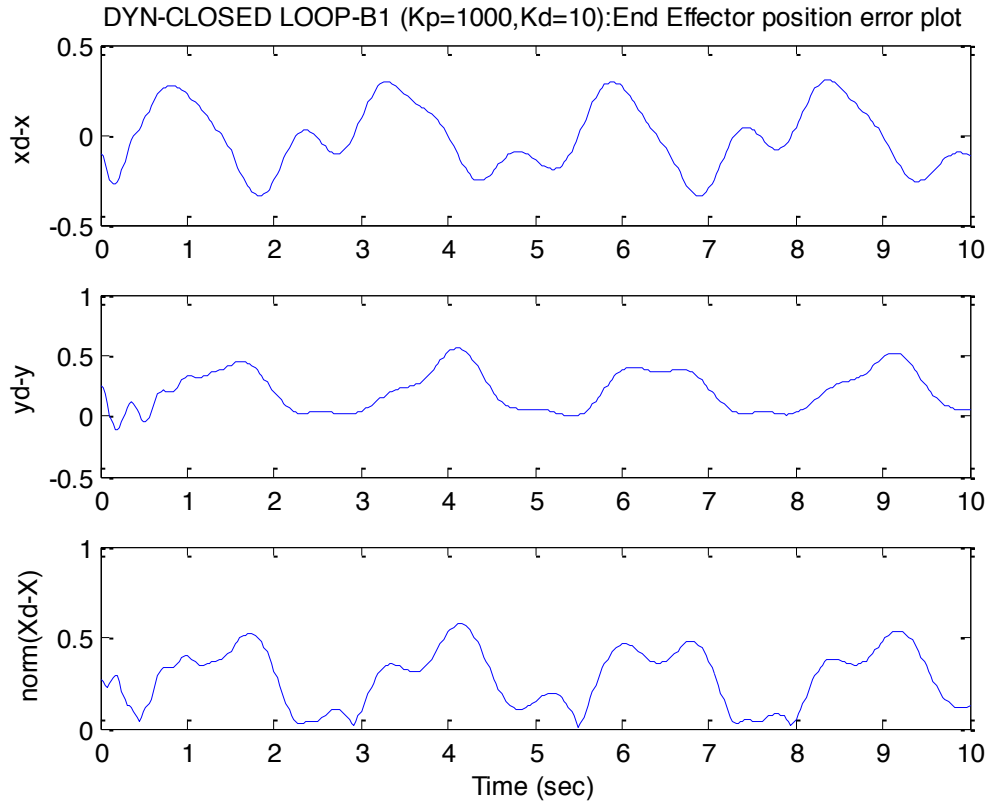

DYN-CLOSED LOOP-B1 (Kp=100,Kd=1):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

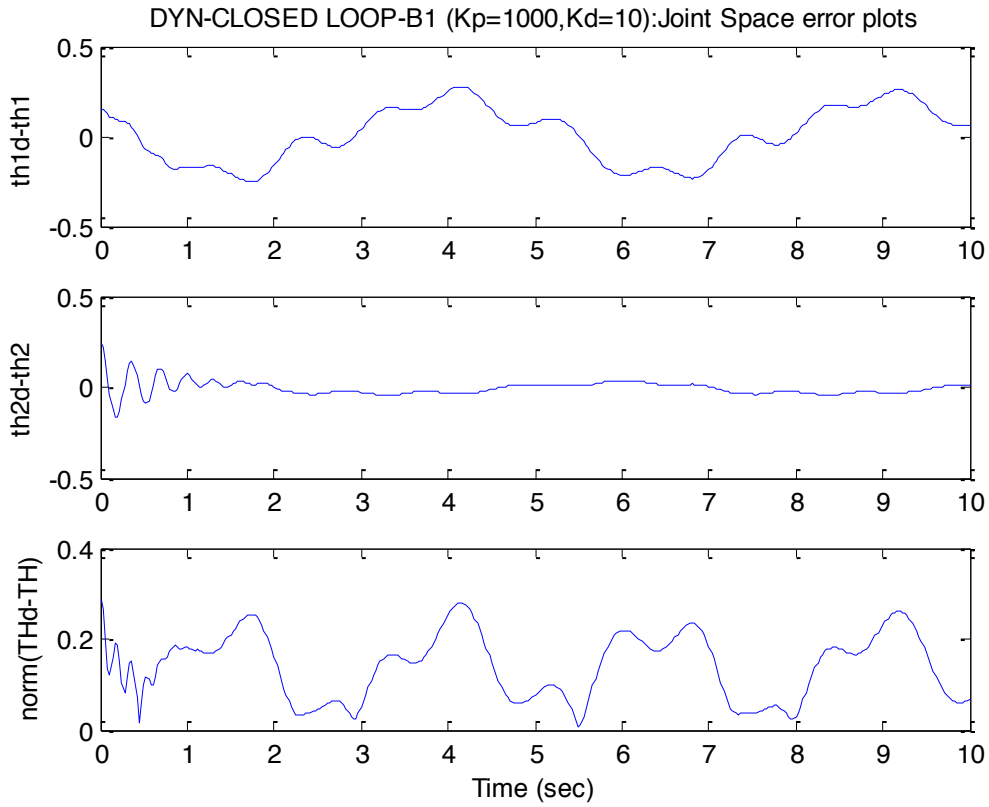

DYN-CLOSED LOOP-B1 (Kp=100,Kd=1)
Torque
Time (sec)
400
300
200
100
0
-100
-200
-300
-400
0
1
2
3
4
5
6
7
8
9
10

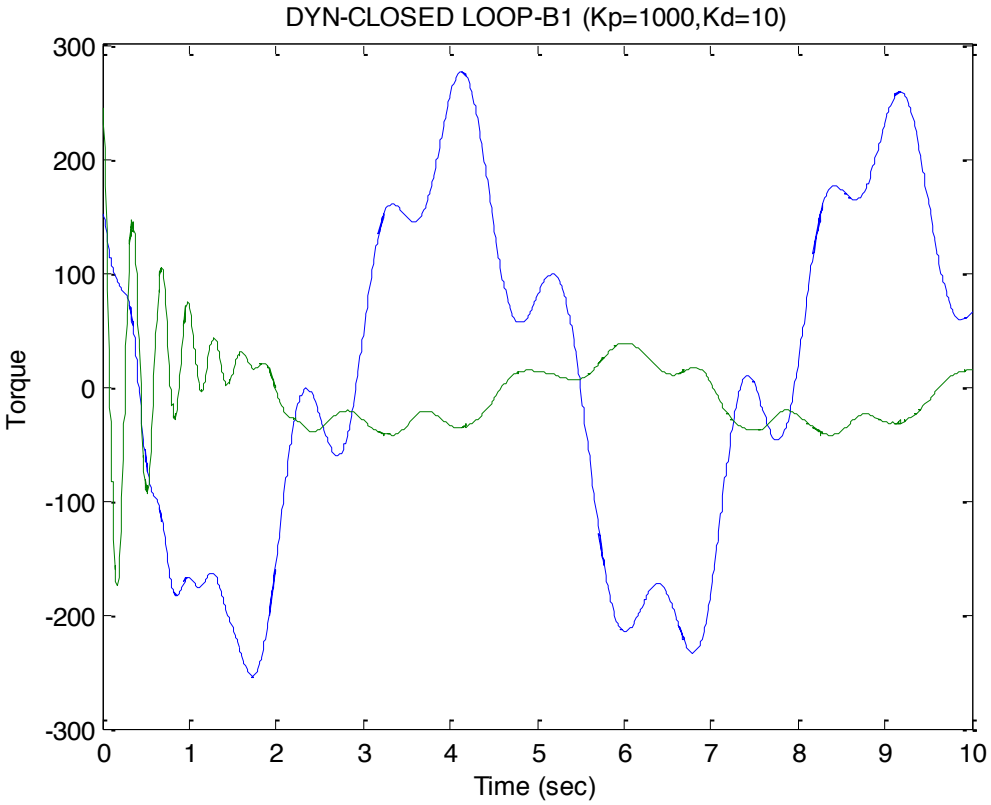
c) K_p=1000(HIGH), K_d=10(LOW)

In this case, we want to study the effect of only K_p on the system. So, K_p has a considerable value while K_d is quite negligible. We see that the output is mainly oscillatory in nature and is hardly damped as time progresses. So, we infer that K_p causes the output to come close to desired and oscillate about it.




DYN-CLOSED LOOP-B1 (Kp=1000,Kd=10):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)


DYN-CLOSED LOOP-B1 (Kp=1000,Kd=10):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)


DYN-CLOSED LOOP-B1 (Kp=1000,Kd=10)
Torque
Time (sec)
300
200
100
0
-100
-200
-300
0
1
2
3
4
5
6
7
8
9
10

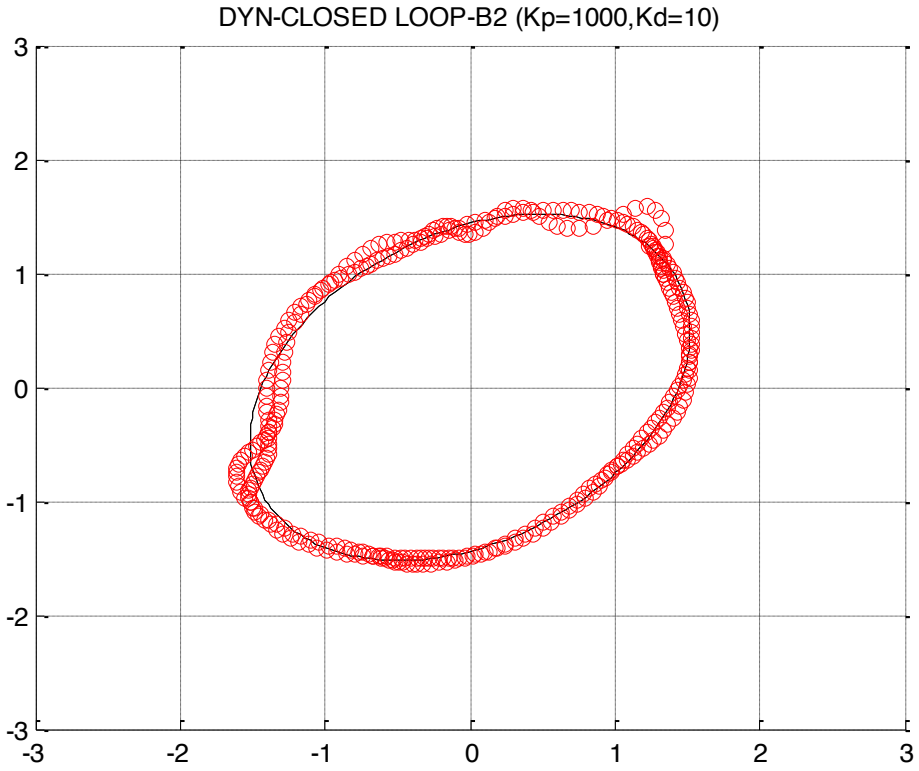
3) B2-GRAVITY+ERROR COMPENSATION

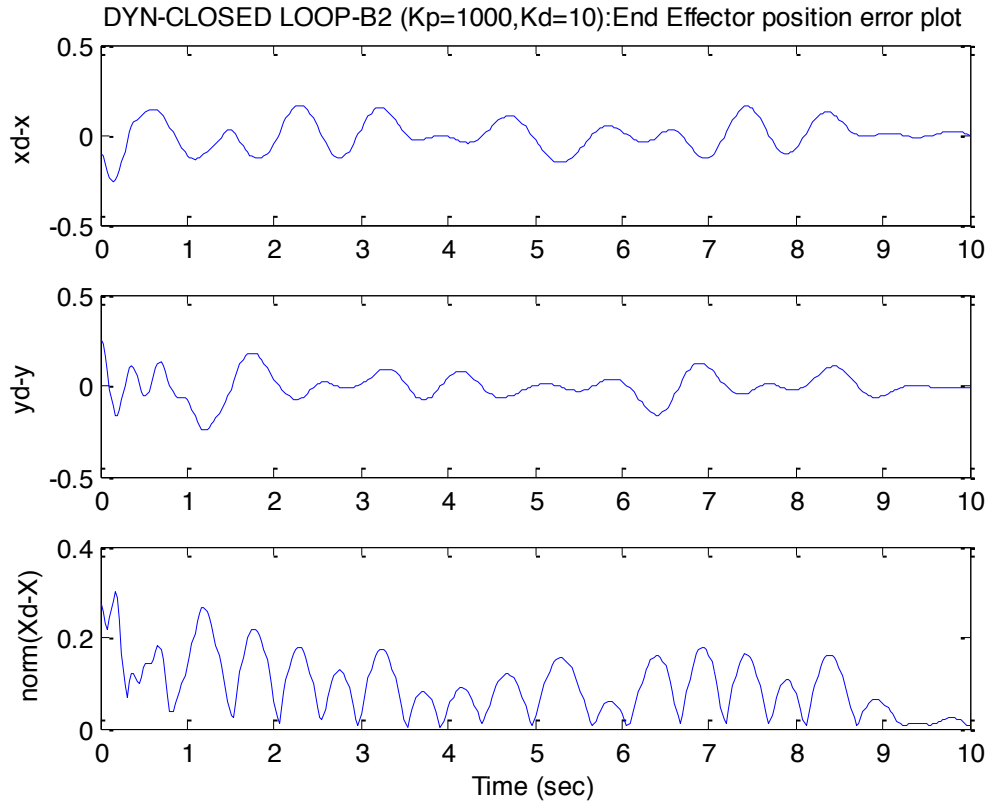
Now, we include gravity terms in the torque to eliminate the level of non-linearity in the system. This should theoretically reduce the errors in the system as it is better accounted for. We start with the last system examined in the previous scenario. The governing equation is:

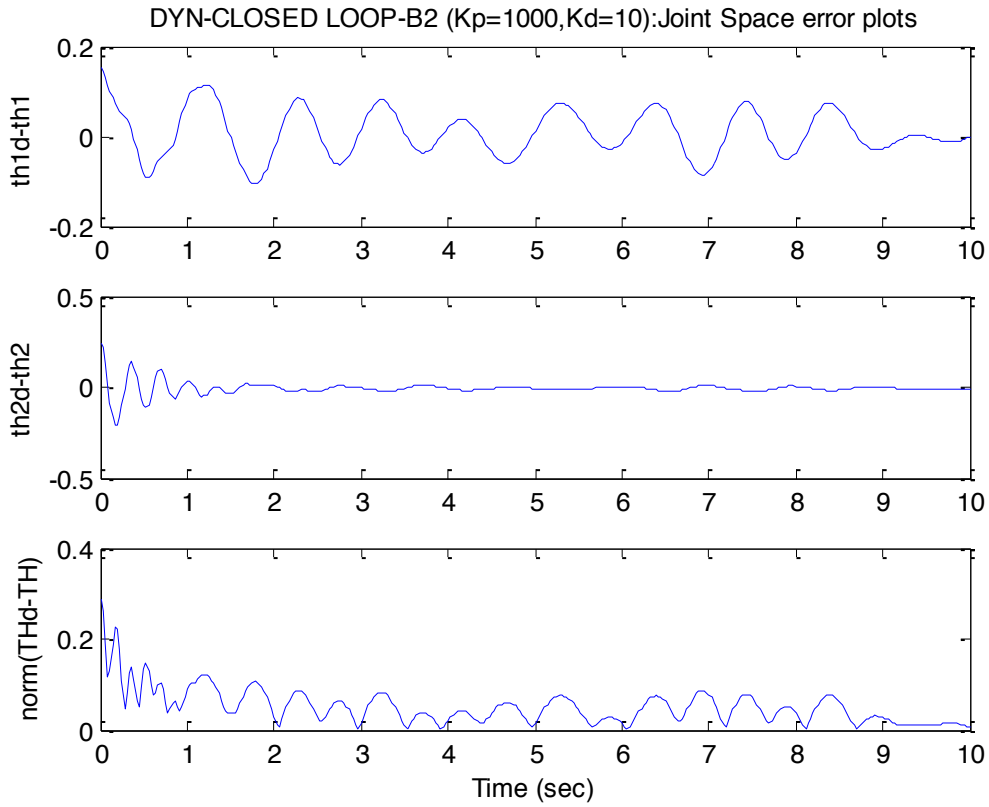
$$\tau = G + K_p\left(\theta^d - \theta\right) + K_d\left(\dot{\theta}^d - \dot{\theta}\right)$$

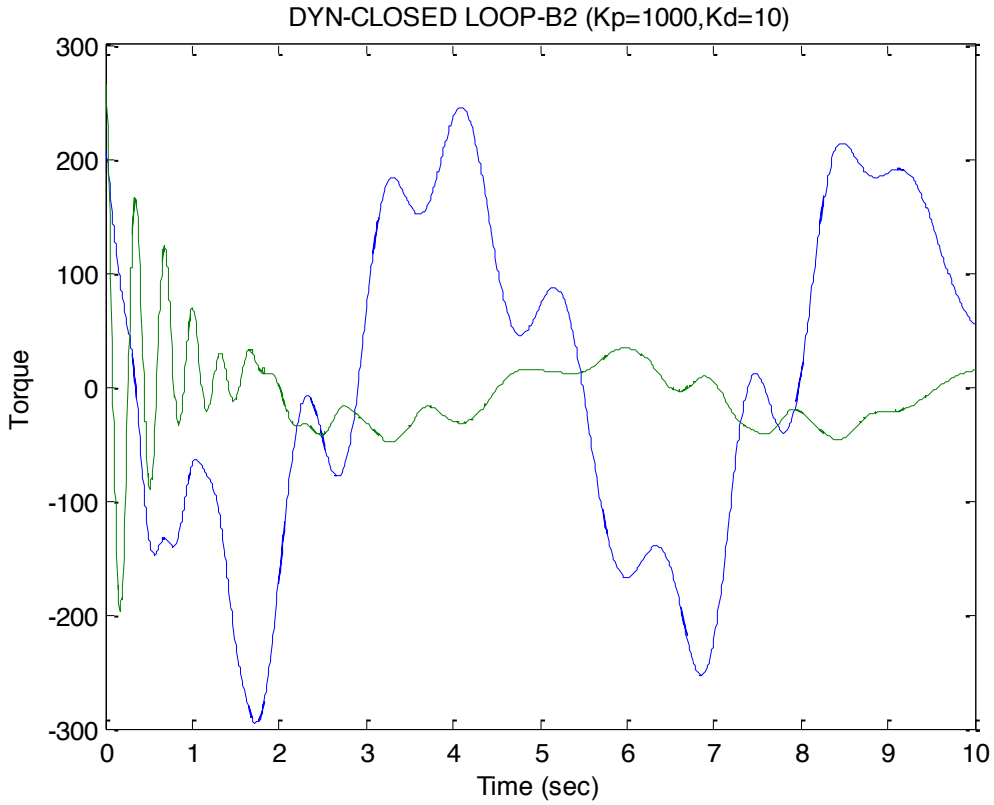
a) K_p=1000(HIGH), K_d=10(LOW)

On comparing with a similar system in the DIRECT ERROR COMPENSATION scenario, we find that errors are considerably reduced. For example, th1 error magnitude has decreased from about 0.3 to 0.18 and the peak torque has reduced from around 280 to around 250. Similar reductions are available in other variables as well. Hence, we can safely infer that providing system information improves system performance.



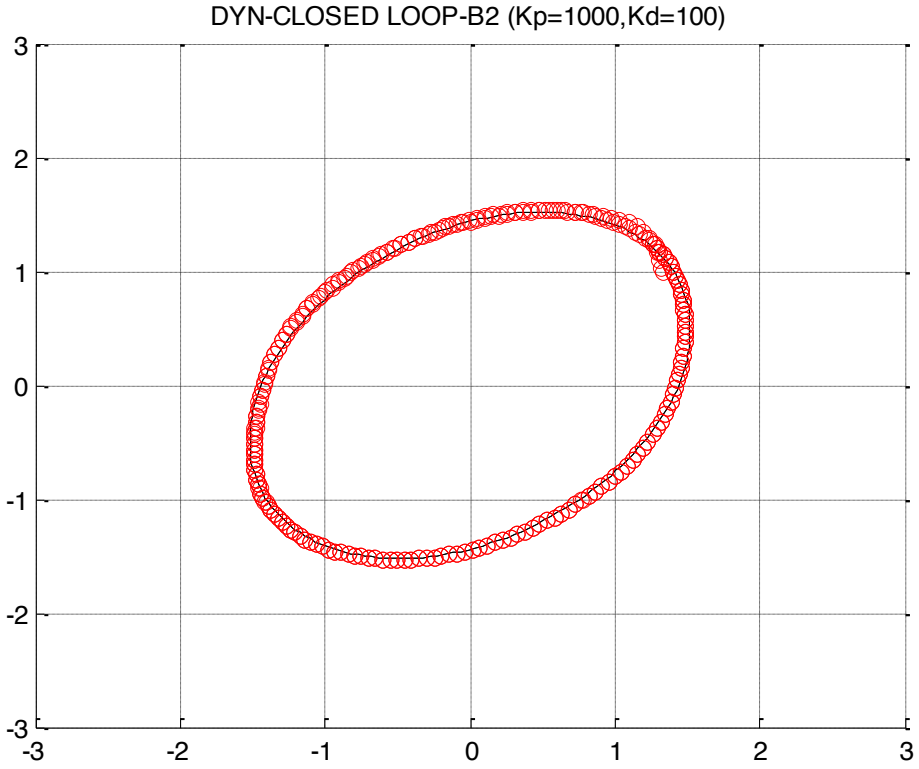

DYN-CLOSED LOOP-B2 (Kp=1000,Kd=10):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)

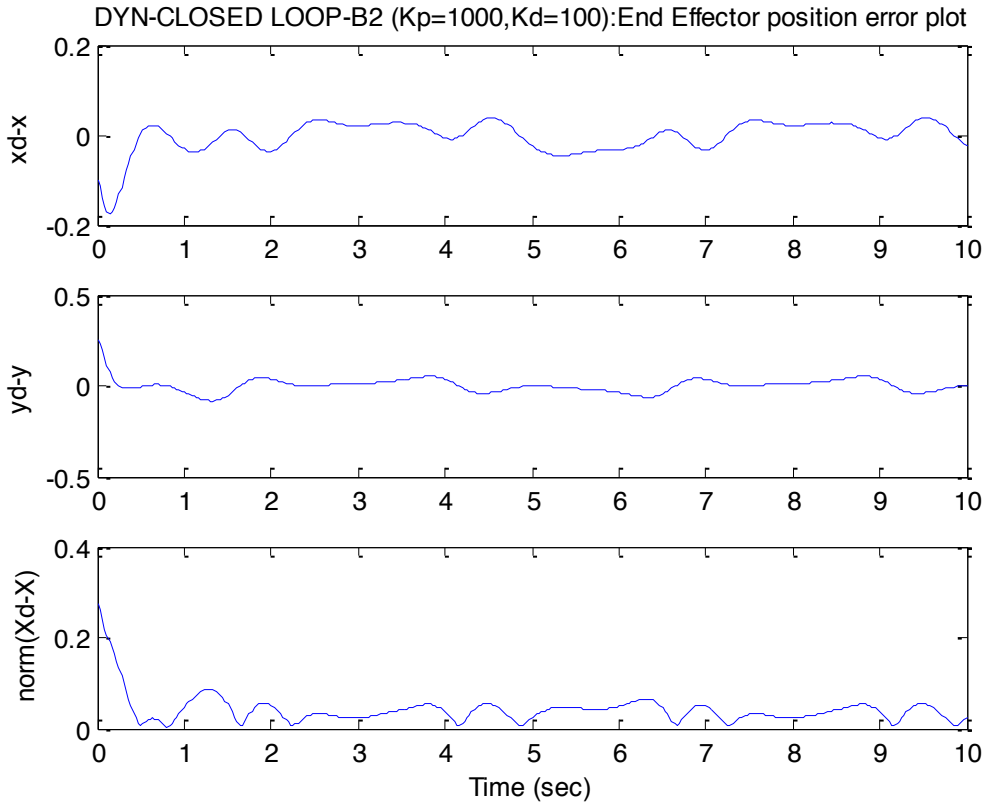

DYN-CLOSED LOOP-B2 (Kp=1000,Kd=10):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)

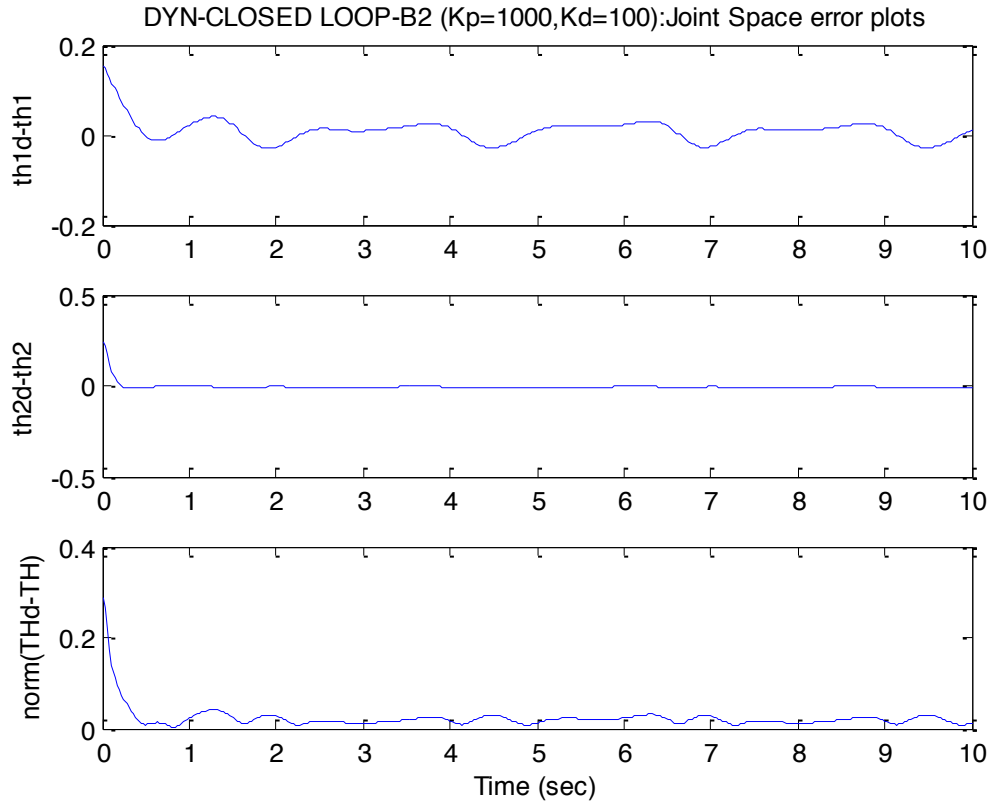

DYN-CLOSED LOOP-B2 (Kp=1000,Kd=10)
Torque
Time (sec)
300
200
100
0
-100
-200
-300
0
1
2
3
4
5
6
7
8
9
10

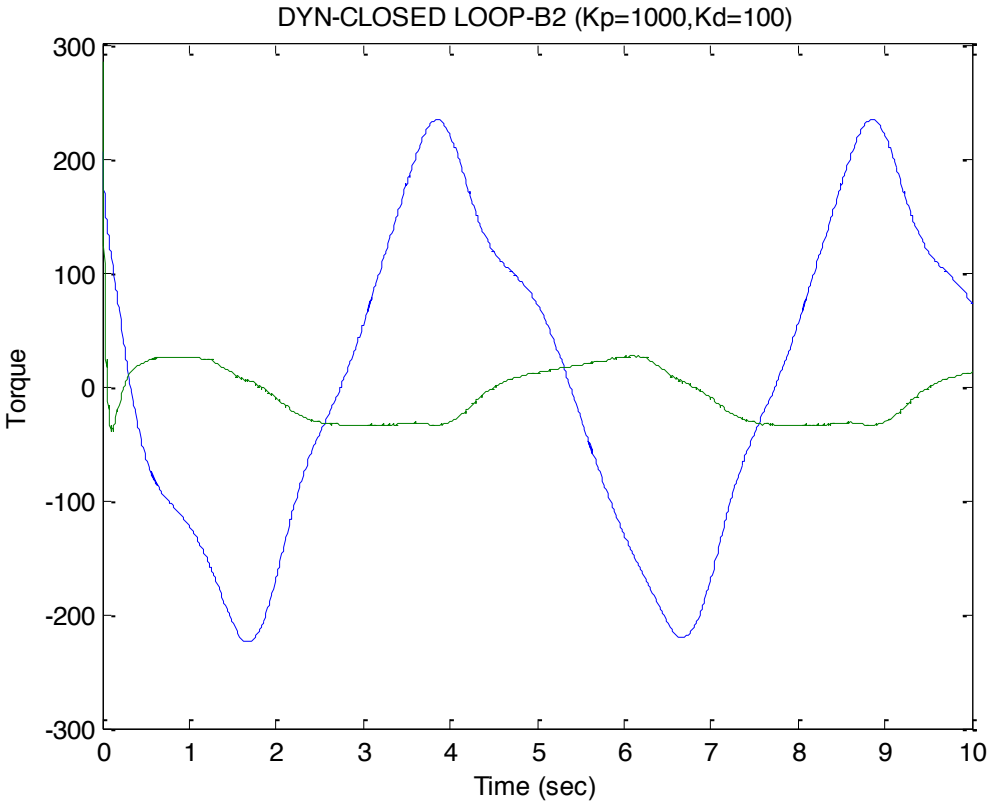
b) K_p=1000(HIGH), K_d=100(HIGH)

To achieve better control, we increase both K_p and K_d and hence better results are achieved as compared to the previous case.






DYN-CLOSED LOOP-B2 (Kp=1000,Kd=100):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)


DYN-CLOSED LOOP-B2 (Kp=1000,Kd=100)
Torque
Time (sec)

4) B3- 'M-V-G'+DIRECT ERROR COMPENSATION (SPRING DAMPER ERROR)

In this case, we provide complete system information and model it so that the error now becomes a simple second degree equation whose damping coefficient and natural frequency are functions of the gains in the system. The governing equations are:

$$\tau = M(\theta)\{\ddot{\theta}^d + K_p(\theta^d - \theta) + K_d(\dot{\theta}^d - \dot{\theta})\} + V + G$$

Considering the original equation for torque,

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta)$$

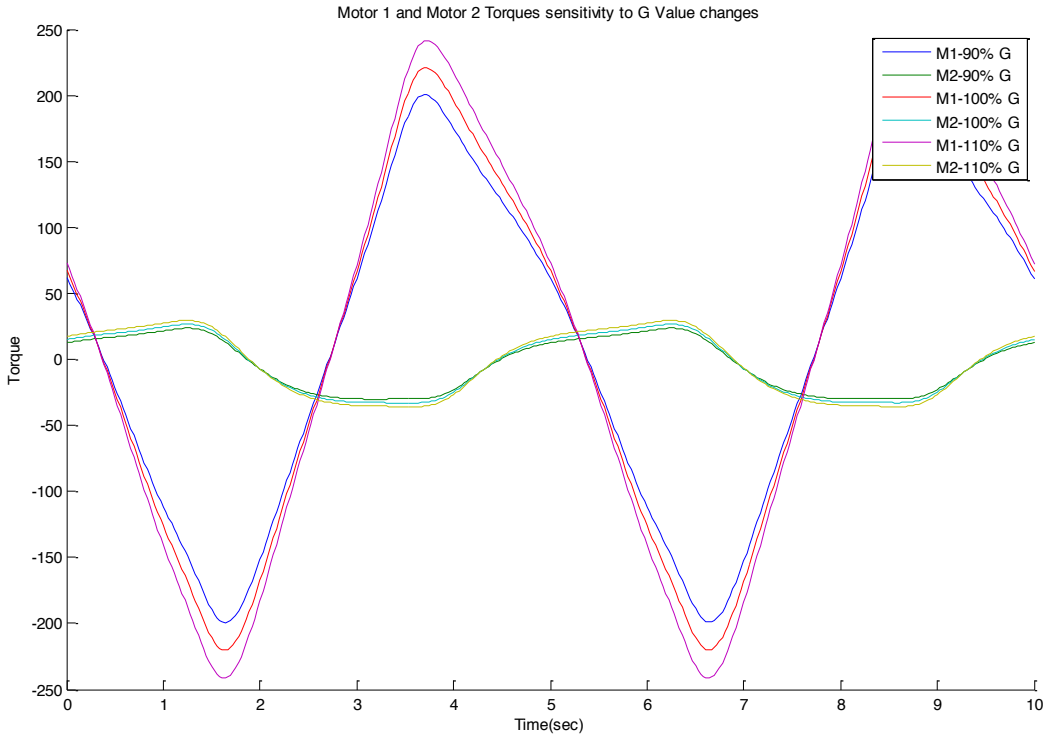
And assuming M and V in both cases almost equal, this equation reduces to

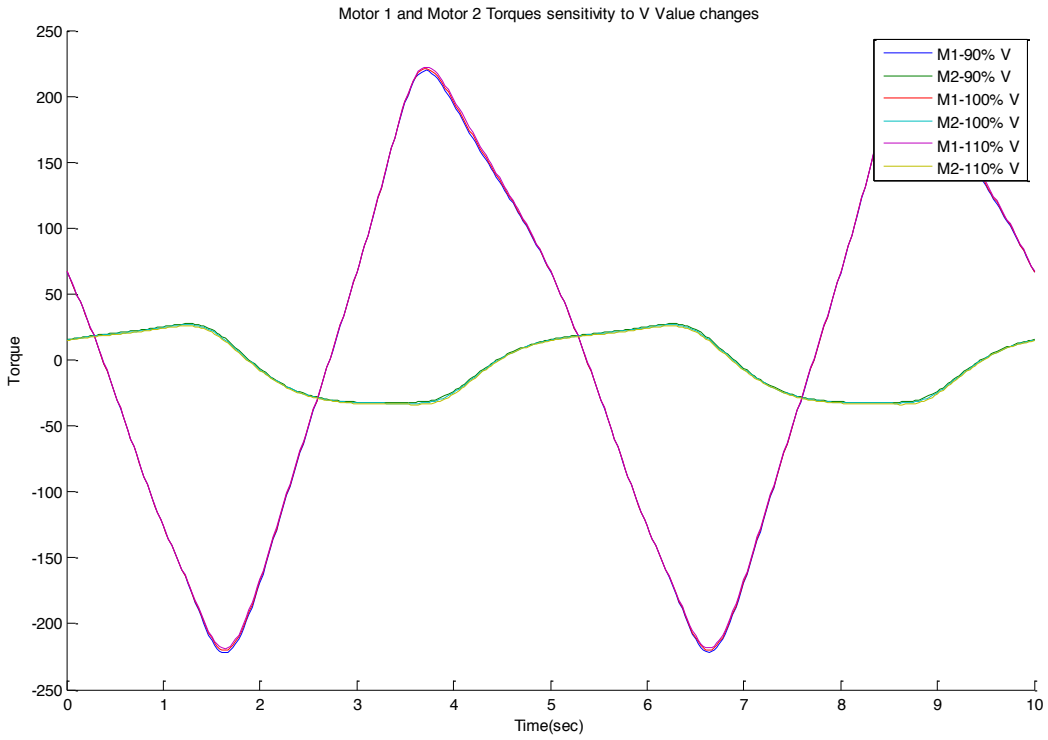
$$\ddot{\theta}^e + K_d\dot{\theta}^e + K_p\theta^e = 0$$

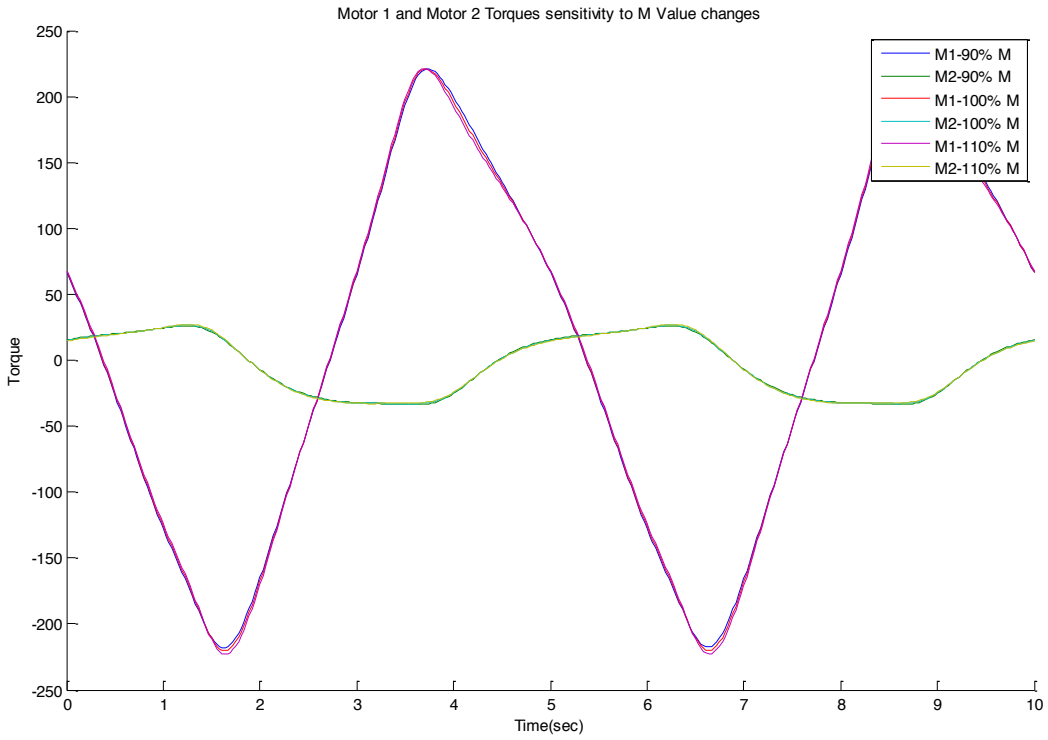
This has an exponential convergence that can be altered as required by tweaking the values of K_p and K_d.

a) Sensitivity analysis

To better understand the system, we tried to determine which component of the equation is significant. The M, V and G values were each tweaked slightly turn by turn and corresponding torque changes were monitored. The results are as shown below. On examining, G turns out to be the major contributor. This is in tandem with the observation that most non-linearity in the system died down in the previous scenario itself when G was accounted for.

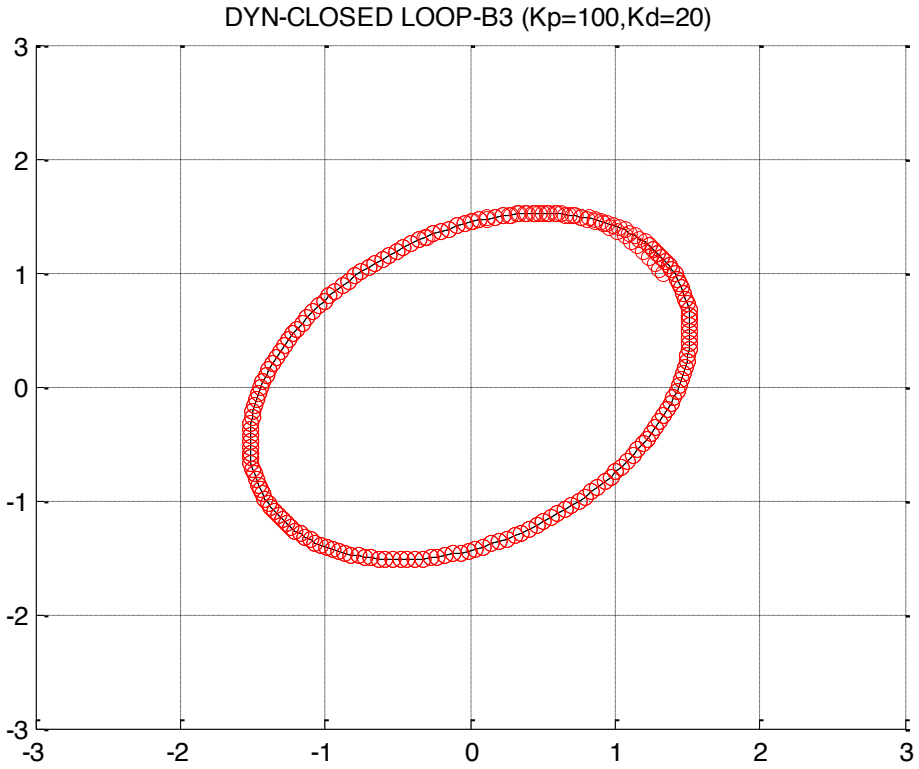


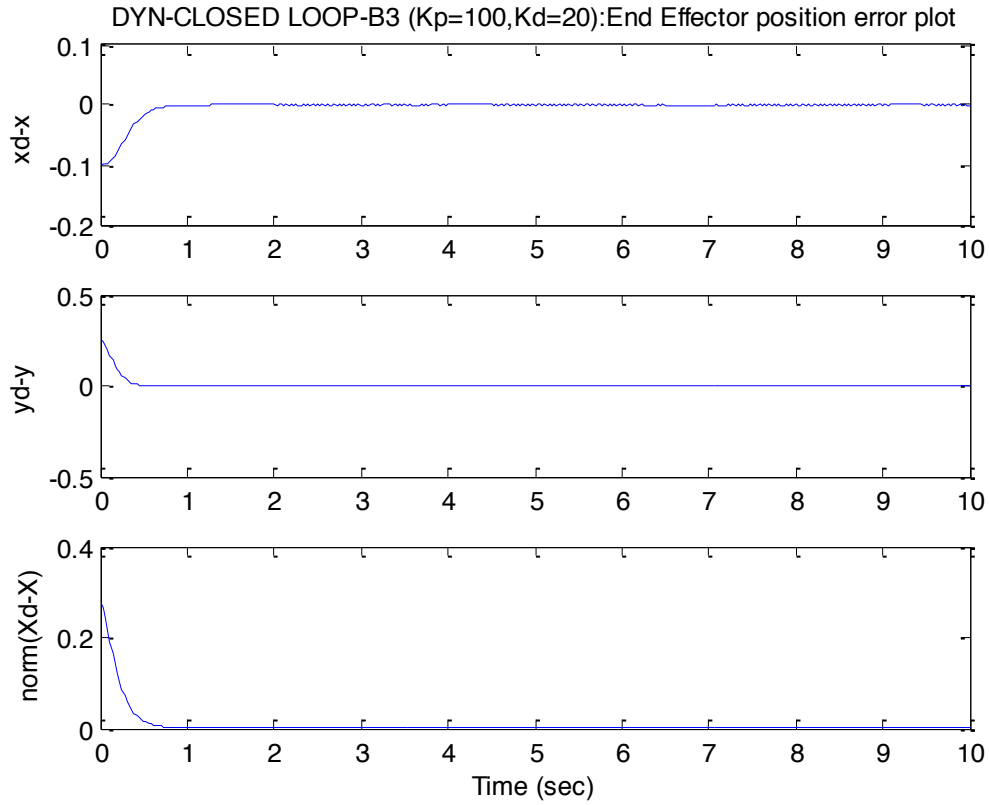

Motor 1 and Motor 2 Torques sensitivity to V Value changes
M1-90% V
M2-90% V
M1-100% V
M2-100% V
M1-110% V
M2-110% V
Torque
Time(sec)

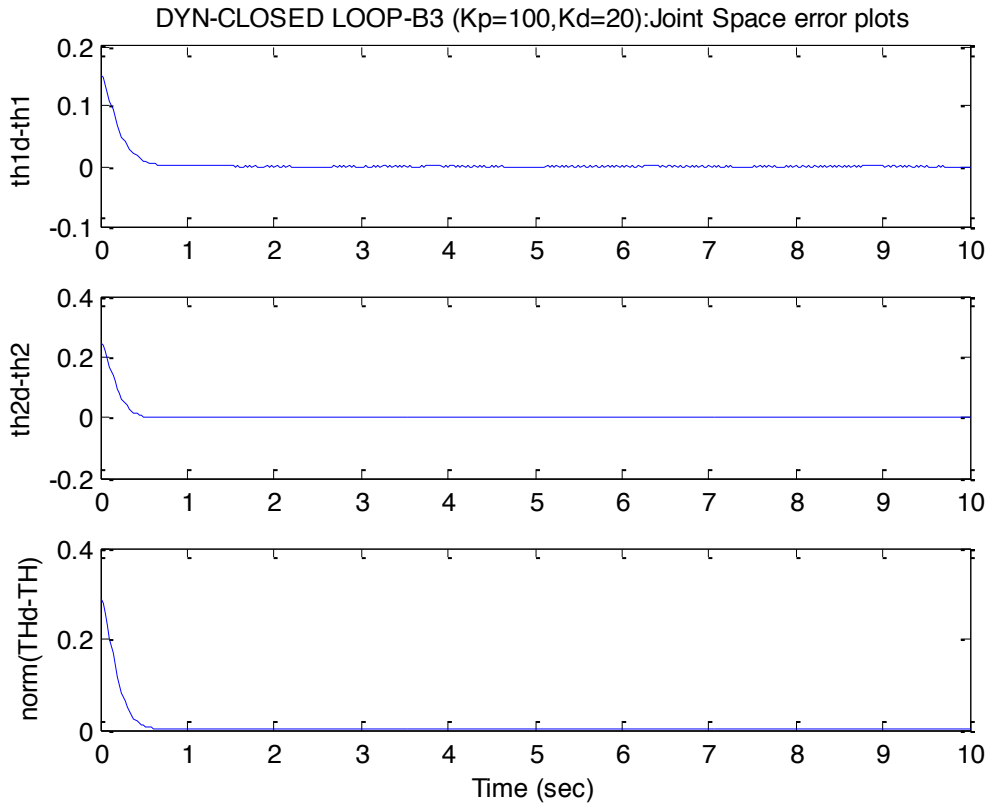

Motor 1 and Motor 2 Torques sensitivity to M Value changes
M1-90% M
M2-90% M
M1-100% M
M2-100% M
M1-110% M
M2-110% M
Torque
Time(sec)

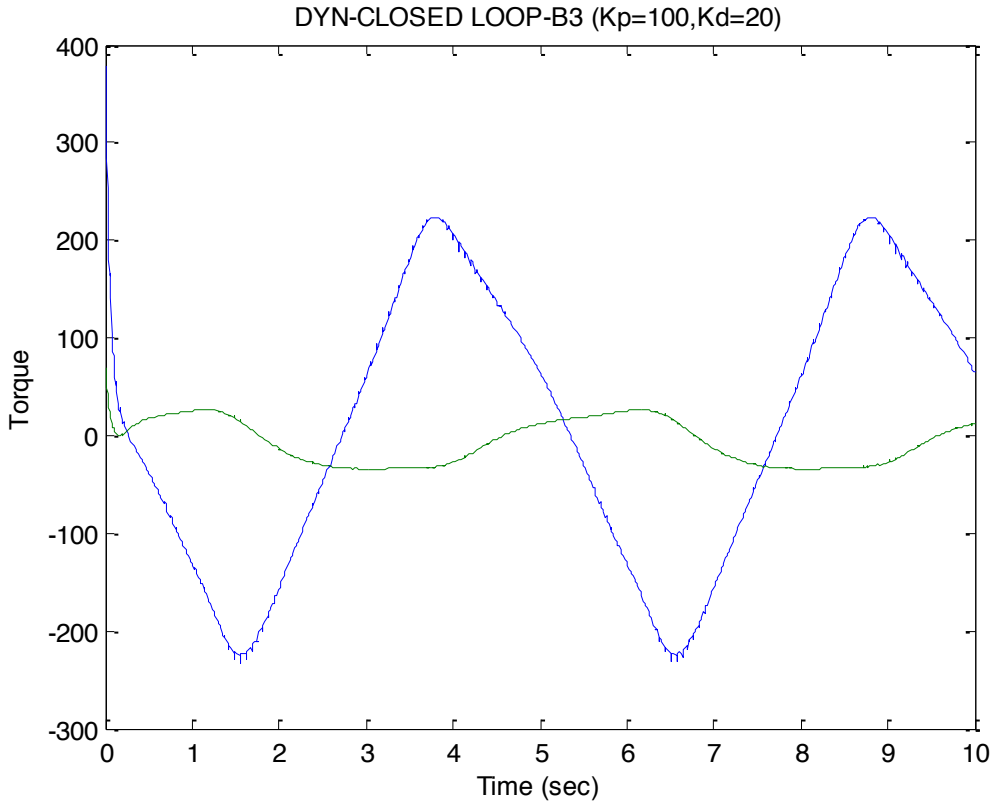
b) Normal parameters

This system is now driven using low values of both K_p and K_d to establish that now minimal level of external control is required as the whole system is accounted for properly and non-linearity is mostly eliminated. This is the most desirable sort of output so far.




DYN-CLOSED LOOP-B3 (Kp=100,Kd=20):End Effector position error plot
xd-x
yd-y
norm(Xd-X)
Time (sec)


DYN-CLOSED LOOP-B3 (Kp=100,Kd=20):Joint Space error plots
th1d-th1
th2d-th2
norm(THd-TH)
Time (sec)



C) CONCLUSION

We have examined a variety of kinematic and dynamic methods for analysis of the two-link parameter system. Each method was derived from the previous one to improve system performance. Although more system information was generally required from one model to the next, it resulted in better control. So, we can conclude that in general, more system information leads to better control.

1) Open Loop Kinematic analysis: Just depended on the desired th and thdot and no feedback control was involved. Error propagated without much aggravation.
2) Joint Space Kinematic Control: Incorporated theta position error as an error signal and improved performance.
3) Task Space Kinematic Control: Incorporated x-y position error as the error signal and further improved performance since $X_e=L*Th_e$.
4) Open Loop Dynamic: Depended on desired th,thdot and thdotdot. Error aggravated due to 'actual position' error propagating temporally in the system. Torque was calculated from the desired joint space data.
5) Direct Error Compensation: Much Better Control. Although lower gains led the system to be unstable, higher gains gave satisfactory results. K_p was found to relate to the oscillatory behavior while K_d affected damping.
6) G+ Direct Error Compensation: Better than the previous case because non-linearity due to gravity terms was eliminated (which was later found to be the major contributor to torque). Torque requirements and errors in variables were reduced.
7) "M-V-G" + Direct Error Compensation: Better than the previous case because non-linearity was almost eliminated. Maximum system information is introduced. Torque requirements and errors are slightly reduced because M and V contributions are found to be small as compared to G terms contributions.

PROGRAMS

1) DYN_MAIN.m

```
%%%%%% Robotics HW4 (Kinematic and Dynamic control and analysis for 2-link serial
chain manipulator)

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah


clc
clear all
close all

%% INITIALIZATION
global rx ry ell_an start_an w Kp Kv iterations thdotdot thdot theta
global error1 error2 error3 error4 error5 error6 error7
global l1 l2 lc1 lc2 j1 j2 m1 m2 g A B rx ry ell_an w Kp Kx Kp1 Kd1 % Given
parameters

%% INPUT PARAMETERS
l1= 2 ;% (m)Length of element 1
lc1= 1 ;%(m) Distance of CM of element 1 from source end
m1= 10;% (kg) Mass of element 1
j1= 3;% (kg-m^2) Moment of Inertia of element 1 about its CM
tau1= 0;% (N-m) External torque applied on element 1

l2= 1;% (m)Length of element 2
lc2= 0.5;% (m)Distance of CM of element 2 from source end
m2= 6;% (kg) Mass of element 2
j2= 2;% (kg-m^2)Moment of Inertia of element 2 about its CM
tau2= 0;% (N-m)External torque applied on element 2

g= 9.81; % (m/sec^2) Acceleration due to gravity

%%

%% DESIRED TRAJECTORY DATA
d2r=pi/180; %degrees to radians
w=72*d2r; % rotational velocity rad/s
rx=1.75; ry=1.25; % ellipse radii
ell_an=45*d2r; % angle of inclination of ellipse
%start_an=45*d2r; % angle of ellipse
A=0;
B=0;

%% INITIAL CONDITIONS
x_0=1.75*cos(45*d2r)+0.1 % initial x in task space
y_0=1.25*sin(45*d2r)+0.1 % initial y in task space

xd_0= -rx*w*sin(w*0)*cos(ell_an)-ry*w*cos(w*0)*sin(ell_an); % initial x-velcity in
task space
```

```
yd_0= -rx*w*sin(w*0)*sin(ell_an)+ry*w*cos(w*0)*cos(ell_an);% initial y-velcity in
task space
% th_0=invbot_new([x_0, y_0]); % initial position in joint space
th_0=invbot([x_0, y_0]); % initial position in joint space
J=[-l1*sin(th_0(1)),-l2*sin(th_0(2));
    l1*cos(th_0(1)), l2*cos(th_0(2))];
thd_0=(inv(J)*[xd_0,yd_0]')'; % initial joint space velocities

%% SIMULATION PARAMETERS
n=360; % total number of points to be simulated
sim_time=10; % 20 second simulation
dt=sim_time/n; % each time step in a 20 second simulation
tspan=0:dt:sim_time;
options = odeset('RelTol',1e-4);

%     case 1 %% Kinematic Simulation: Open Loop
%     case 2    %% Kinematic Simulation: Closed Loop-Joint Space Kp=1,1
%     case 3    %% Kinematic Simulation: Closed Loop-Joint Space Kp=100,100
%     case 4    %% Kinematic Simulation: Closed Loop-Joint Space Kp=100,1
%     case 5 %% Kinematic Simulation: Closed Loop-Task Space Kx=1,1
%     case 6 %% Kinematic Simulation: Closed Loop-Task Space Kx= 100,100
%     case 7 %% Kinematic Simulation: Closed Loop-Task Space Kx= 100,10
%     case 8    %% Dynamic Simulation: Open Loop
%     case 9 %% Dynamic Closed Loop (B1)
%     case 10 %% Dynamic Closed Loop (B2)
%     case 11 %% Dynamic Closed Loop (B3)

TRIG= 11;
switch (TRIG)
    case 1 %% Kinematic Simulation: Open Loop
        global itr Q1 t1
        itr=0;
        txt1=['KIN_OPEN LOOP'];
        [t,Y1]=ode45(@KIN_OL,tspan,[th_0],options);
        plotbot_js_new(tspan,Y1,1,txt1);
        ERROR_PLOT_1(t,Y1,1,txt1);

        %%%%% To find torque from simulated Theta
        tau_sim = FIND_TAU_SIM(tspan,Y1)
        figure()
        plot(tspan,tau_sim);
        ylabel('Torque');
        xlabel('Time(sec)');
        title('Torque from simulated data');

    case 2    %% Kinematic Simulation: Closed Loop-Joint Space Kp=1,1
        global itr Q2 t2
        itr=0;
        Kp= [1 0;0 1];
        txt1=['KIN-CLOSED LOOP-JS (Kp=',num2str(Kp(1,1)),',',num2str(Kp(2,2)),')'];
        [t,Y2]=ode45(@KIN_CLJS,tspan,[th_0],options);
        plotbot_js_new(tspan,Y2,2,txt1);
        ERROR_PLOT_1(t,Y2,2,txt1);

    case 3    %% Kinematic Simulation: Closed Loop-Joint Space Kp=100,100
        global itr Q2 t2
        itr=0;
        Kp= [100 0;0 100];
```

```
        txt1=['KIN-CLOSED LOOP-JS (Kp=',num2str(Kp(1,1)),',',num2str(Kp(2,2)),')'];
        [t,Y2]=ode45(@KIN_CLJS,tspan,[th_0],options);
        plotbot_js_new(tspan,Y2,2,txt1);
        ERROR_PLOT_1(t,Y2,2,txt1);

    case 4    %% Kinematic Simulation: Closed Loop-Joint Space Kp=100,1
        global itr Q2 t2
        itr=0;
        Kp= [100 0;0 1];
        txt1=['KIN-CLOSED LOOP-JS (Kp=',num2str(Kp(1,1)),',',num2str(Kp(2,2)),')'];
        [t,Y2]=ode45(@KIN_CLJS,tspan,[th_0],options);
        plotbot_js_new(tspan,Y2,2,txt1);
        ERROR_PLOT_1(t,Y2,2,txt1);

    case 5 %% Kinematic Simulation: Closed Loop-Task Space Kx=1,1
        global itr Q3 t3
        itr=0;
        Kx= [1 0;0 1];
        txt1=['KIN-CLOSED LOOP-TS (Kx=',num2str(Kx(1,1)),',',num2str(Kx(2,2)),')'];
        [t,Y3]=ode45(@KIN_CLTS,tspan,[th_0],options);
        plotbot_js_new(tspan,Y3,3,txt1);
        ERROR_PLOT_1(t,Y3,3,txt1);

    case 6 %% Kinematic Simulation: Closed Loop-Task Space Kx= 100,100
        global itr Q3 t3
        itr=0;
        Kx= [100 0;0 100];
        txt1=['KIN-CLOSED LOOP-TS (Kx=',num2str(Kx(1,1)),',',num2str(Kx(2,2)),')'];
        [t,Y3]=ode45(@KIN_CLTS,tspan,[th_0],options);
        plotbot_js_new(tspan,Y3,3,txt1);
        ERROR_PLOT_1(t,Y3,3,txt1);

    case 7 %% Kinematic Simulation: Closed Loop-Task Space Kx= 100,10
        global itr Q3 t3
        itr=0;
        Kx= [100 0;0 10];
        txt1=['KIN-CLOSED LOOP-TS (Kx=',num2str(Kx(1,1)),',',num2str(Kx(2,2)),')'];
        [t,Y3]=ode45(@KIN_CLTS,tspan,[th_0],options);
        plotbot_js_new(tspan,Y3,3,txt1);
        ERROR_PLOT_1(t,Y3,3,txt1);

%% DYNAMIC SIMULATION
    case 8    %% Dynamic Simulation: Open Loop
        global itr Q4 t4 Tau1
        itr=0;
        txt1=['DYN-OPEN LOOP'];
        [t,Y4]=ode45(@DYN_OL,tspan,[th_0,thd_0],options);
        plotbot_js_new(t,Y4,4,txt1);
        ERROR_PLOT_2(t,Y4,4,txt1);
        figure
        plot(t4,Tau1);
        xlabel('Time (sec)');
        ylabel('Torque');
        title(txt1);

    case 9 %% Dynamic Closed Loop (B1)
        global itr Q5 t5 Tau2
        itr=0;
```

```
        Kp1= [1000, 0;0, 1000];%%%Dynamic analysis
        Kd1= [10, 0;0, 10];%%%Dynamic analysis
        txt1=['DYN-CLOSED LOOP-B1 (Kp=',num2str(Kp1(1)),',Kd=',num2str(Kd1(1)),')'];
        [t,Y5]=ode45(@DYN_CL_B1,tspan,[th_0,thd_0],options);
        plotbot_js_new(tspan,Y5,5,txt1);
        ERROR_PLOT_2(t,Y5,5,txt1);
        figure
        plot(t5,Tau2);
        xlabel('Time (sec)');
        ylabel('Torque');
        title(txt1);
    case 10 %% Dynamic Closed Loop (B2)
        global itr Q6 t6 Tau3
        itr=0;

        Kp1= [1000, 0;0, 1000];%%%Dynamic analysis
        Kd1= [100, 0;0, 100];%%%Dynamic analysis
        txt1 = ['DYN-CLOSED LOOP-B2
(Kp=',num2str(Kp1(1)),',Kd=',num2str(Kd1(1)),')'];
        [t,Y6]=ode45(@DYN_CL_B2,tspan,[th_0,thd_0],options);
        plotbot_js_new(tspan,Y6,6,txt1);
        ERROR_PLOT_2(t,Y6,6,txt1);
        figure
        plot(t6,Tau3);
        xlabel('Time (sec)');
        ylabel('Torque');
        title(txt1);
    case 11 %% Dynamic Closed Loop (B3)
        global itr Q7 t7 Tau4
        itr=0;
        Kp1= [100, 0;0, 100];%%%Dynamic analysis
        Kd1= [20, 0;0, 20];%%%Dynamic analysis
        txt1=['DYN-CLOSED LOOP-B3 (Kp=',num2str(Kp1(1)),',Kd=',num2str(Kd1(1)),')'];
        [t,Y7]=ode45(@DYN_CL_B3,tspan,[th_0,thd_0],options);
        plotbot_js_new(tspan,Y7,7,txt1);
        ERROR_PLOT_2(t,Y7,7,txt1);
        figure
        plot(t7,Tau4);
        xlabel('Time (sec)');
        ylabel('Torque');
        title(txt1);
    otherwise
        'ENTER CORRECT TRIGGER VALUE'
end
```

2) DYNAMIC ANALYSIS : Open Loop

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [dX]= DYN_OL(t,X)


global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given
parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8
global Tau1 Tau2 Tau3 Tau4

th1=X(1);
th2=X(2);
th1d=X(3);
th2d=X(4);


%% CREATING IMPORTANT MATRICES IN THE GOVERNING EQUATION

%% Acquiring desired theta values for points exactly on the ellipse%%%%%%%
THETA_DES=TH_DES_INFO(t,[th1,th2]);
th1_des=THETA_DES(1);
th2_des=THETA_DES(2);
th1d_des=THETA_DES(3);
th2d_des=THETA_DES(4);
th1dd_des=THETA_DES(5);
th2dd_des=THETA_DES(6);

s1_des=sin(th1_des);
s2_des=sin(th2_des);
c1_des=cos(th1_des);
c2_des=cos(th2_des);
c21_des=cos(th2_des-th1_des);
s21_des=sin(th2_des-th1_des);

%% Dynamic control matrices : Desired
Mdes = [ j1+m2*l1^2, m2*l1*lc2*c21_des;
      m2*l1*lc2*c21_des, j2+m2*lc2^2];
Vdes = [0, -m2*l1*lc2*s21_des;
    m2*l1*lc2*s21_des, 0];
Gdes = [m1*g*lc1*c1_des+m2*g*l1*c1_des;
    m2*g*lc2*c2_des];

%% Control Torque (based on ideal situation)
Tau_des= Mdes*[th1dd_des;th2dd_des] + Vdes*[th1d_des^2;th2d_des^2] + Gdes; %% Control
Torque

%% Simulation Data
s1=sin(th1);
s2=sin(th2);
```

```
c1=cos(th1);
c2=cos(th2);
c21=cos(th2-th1);
s21=sin(th2-th1);

%%%%%% Dynamic control matrices
M = [ j1+m2*l1^2, m2*l1*lc2*c21;
      m2*l1*lc2*c21, j2+m2*lc2^2];
V = [0, -m2*l1*lc2*s21;
    m2*l1*lc2*s21, 0];
G = [m1*g*lc1*c1+m2*g*l1*c1;
    m2*g*lc2*c2];
%%%%%%%%%%%%%%%%%%%%%%%%

%% Generating differential
% Tau_des = M*[th1dd_des;th2dd_des] + V*[th1d_des^2;th2d_des^2] + Gdes; %% Control
Torque
[THDD]=inv(M)*(Tau_des- V*[th1d^2;th2d^2]-G); %%% Deriving angular acc
% [THDD]=inv(M-Mdes)*(Vdes-V*[th1d^2;th2d^2] + (Gdes-G)); %%% Deriving angular acc
%% Simulation Update
dX(1,1)= th1d;
dX(2,1)= th2d;
dX(3,1)= THDD(1);
dX(4,1)= THDD(2);
%% Storage
itr=itr+1;
t4(itr)=t;
size(th1);
size(th2);
Q4(itr,1) = [th1];
Q4(itr,2) = [th2];
Q4(itr,3) = [th1d];
Q4(itr,4) = [th2d];
Q4(itr,5) = THDD(1);
Q4(itr,6) = THDD(2);
Tau1(itr,1:2) = Tau_des;
```

3) DYNAMIC ANALYSIS : CLOSED LOOP B1

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [dX]= DYN_CL_B1(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given
parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8
global Tau1 Tau2 Tau3 Tau4

th1=X(1);
th2=X(2);
th1d=X(3);
th2d=X(4);


%% CREATING IMPORTANT MATRICES IN THE GOVERNING EQUATION

%% Acquiring desired theta values for points exactly on the ellipse%%%%%%%
THETA_DES=TH_DES_INFO(t,[th1,th2]);
th1_des=THETA_DES(1);
th2_des=THETA_DES(2);
th1d_des=THETA_DES(3);
th2d_des=THETA_DES(4);
th1dd_des=THETA_DES(5);
th2dd_des=THETA_DES(6);

s1_des=sin(th1_des);
s2_des=sin(th2_des);
c1_des=cos(th1_des);
c2_des=cos(th2_des);
c21_des=cos(th2_des-th1_des);
s21_des=sin(th2_des-th1_des);

%% Dynamic control matrices : Desired
Mdes = [ j1+m2*l1^2, m2*l1*lc2*c21_des;
      m2*l1*lc2*c21_des, j2+m2*lc2^2];
Vdes = [0, -m2*l1*lc2*s21_des;
    m2*l1*lc2*s21_des, 0];
Gdes = [m1*g*lc1*c1_des+m2*g*l1*c1_des;
    m2*g*lc2*c2_des];

%% Control Torque (based on ideal situation)
% Tau_des1= Mdes*[th1dd_des;th2dd_des] + Vdes*[th1d_des^2;th2d_des^2]+Gdes; %%
Control Torque
Tau_des= Kp1*([th1_des;th2_des]-[th1;th2])+Kd1*([th1d_des;th2d_des]-[th1d;th2d]); %%
Control Torque

%% Simulation Data
s1=sin(th1);
```

```
s2=sin(th2);
c1=cos(th1);
c2=cos(th2);
c21=cos(th2-th1);
s21=sin(th2-th1);

%%%%%% Dynamic control matrices
M = [ j1+m2*l1^2, m2*l1*lc2*c21;
      m2*l1*lc2*c21, j2+m2*lc2^2];
V = [0, -m2*l1*lc2*s21;
    m2*l1*lc2*s21, 0];
G = [m1*g*lc1*c1+m2*g*l1*c1;
    m2*g*lc2*c2];
%%%%%%%%%%%%%%%%%%%%%%%%

%% Generating differential
[THDD]=inv(M)*(Tau_des- V*[th1d^2;th2d^2]-G); %%% Deriving angular acc
%% Simulation Update
dX(1,1)= th1d;
dX(2,1)= th2d;
dX(3,1)= THDD(1);
dX(4,1)= THDD(2);
%% Storage
itr=itr+1;
t5(itr)=t;
Q5(itr,1) = [th1];
Q5(itr,2) = [th2];
Q5(itr,3) = [th1d];
Q5(itr,4) = [th2d];
Q5(itr,5) = THDD(1);
Q5(itr,6) = THDD(2);
Tau2(itr,:) = Tau_des;
```

4) DYNAMIC ANALYSIS : CLOSED LOOP B2

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [dX]= DYN_CL_B2(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given
parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8
global Tau1 Tau2 Tau3 Tau4

th1=X(1);
th2=X(2);
th1d=X(3);
th2d=X(4);


%% CREATING IMPORTANT MATRICES IN THE GOVERNING EQUATION

%% Acquiring desired theta values for points exactly on the ellipse%%%%%%%
THETA_DES=TH_DES_INFO(t,[th1,th2]);
th1_des=THETA_DES(1);
th2_des=THETA_DES(2);
th1d_des=THETA_DES(3);
th2d_des=THETA_DES(4);
th1dd_des=THETA_DES(5);
th2dd_des=THETA_DES(6);

s1_des=sin(th1_des);
s2_des=sin(th2_des);
c1_des=cos(th1_des);
c2_des=cos(th2_des);
c21_des=cos(th2_des-th1_des);
s21_des=sin(th2_des-th1_des);

%% Dynamic control matrices : Desired
Mdes = [ j1+m2*l1^2, m2*l1*lc2*c21_des;
      m2*l1*lc2*c21_des, j2+m2*lc2^2];
Vdes = [0, -m2*l1*lc2*s21_des;
    m2*l1*lc2*s21_des, 0];
Gdes = [m1*g*lc1*c1_des+m2*g*l1*c1_des;
    m2*g*lc2*c2_des];

%% Control Torque (based on ideal situation)
% Tau_des1= Mdes*[th1dd_des;th2dd_des] + Vdes*[th1d_des^2;th2d_des^2]+Gdes; %%
Control Torque
Tau_des= Gdes + Kp1*([th1_des;th2_des]-[th1;th2])+Kd1*([th1d_des;th2d_des]-
[th1d;th2d]); %% Control Torque

%% Simulation Data
s1=sin(th1);
```

```
s2=sin(th2);
c1=cos(th1);
c2=cos(th2);
c21=cos(th2-th1);
s21=sin(th2-th1);

%%%%%% Dynamic control matrices
M = [ j1+m2*l1^2, m2*l1*lc2*c21;
      m2*l1*lc2*c21, j2+m2*lc2^2];
V = [0, -m2*l1*lc2*s21;
    m2*l1*lc2*s21, 0];
G = [m1*g*lc1*c1+m2*g*l1*c1;
    m2*g*lc2*c2];
%%%%%%%%%%%%%%%%%%%%%%%%

%% Generating differential
[THDD]=inv(M)*(Tau_des- V*[th1d^2;th2d^2]-G); %%% Deriving angular acc
%% Simulation Update
dX(1,1)= th1d;
dX(2,1)= th2d;
dX(3,1)= THDD(1);
dX(4,1)= THDD(2);
%% Storage
itr=itr+1;
t6(itr)=t;
Q6(itr,1) = [th1];
Q6(itr,2) = [th2];
Q6(itr,3) = [th1d];
Q6(itr,4) = [th2d];
Q6(itr,5) = THDD(1);
Q6(itr,6) = THDD(2);
Tau3(itr,:)= Tau_des;
```

5) DYNAMIC ANALYSIS : DYNAMIC ANALYSIS CLOSED LOOP B3

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah


function [dX]= DYN_CL_B3(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given
parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8
global Tau1 Tau2 Tau3 Tau4

th1=X(1);
th2=X(2);
th1d=X(3);
th2d=X(4);


%% CREATING IMPORTANT MATRICES IN THE GOVERNING EQUATION

%% Acquiring desired theta values for points exactly on the ellipse%%%%%%%
THETA_DES=TH_DES_INFO(t,[th1,th2]);
th1_des=THETA_DES(1);
th2_des=THETA_DES(2);
th1d_des=THETA_DES(3);
th2d_des=THETA_DES(4);
th1dd_des=THETA_DES(5);
th2dd_des=THETA_DES(6);

s1_des=sin(th1_des);
s2_des=sin(th2_des);
c1_des=cos(th1_des);
c2_des=cos(th2_des);
c21_des=cos(th2_des-th1_des);
s21_des=sin(th2_des-th1_des);

%% Dynamic control matrices : Desired
Mdes = [ j1+m2*l1^2, m2*l1*lc2*c21_des;
      m2*l1*lc2*c21_des, j2+m2*lc2^2];
Vdes = [0, -m2*l1*lc2*s21_des;
    m2*l1*lc2*s21_des, 0];
Gdes = [m1*g*lc1*c1_des+m2*g*l1*c1_des;
    m2*g*lc2*c2_des];

%% Control Torque (based on ideal situation)
% Tau_des1= Mdes*[th1dd_des;th2dd_des] + Vdes*[th1d_des^2;th2d_des^2]+Gdes; %%
Control Torque
Tau_des= Gdes+ Vdes*[th1d_des^2;th2d_des^2] + Mdes*( [th1dd_des;th2dd_des] +
Kp1*([th1_des;th2_des]-[th1;th2]) + Kd1*([th1d_des;th2d_des]-[th1d;th2d])); %%
Control Torque
```

```
%% Simulation Data
s1=sin(th1);
s2=sin(th2);
c1=cos(th1);
c2=cos(th2);
c21=cos(th2-th1);
s21=sin(th2-th1);

%%%%%% Dynamic control matrices
M = [ j1+m2*l1^2, m2*l1*lc2*c21;
      m2*l1*lc2*c21, j2+m2*lc2^2];
V = [0, -m2*l1*lc2*s21;
    m2*l1*lc2*s21, 0];
G = [m1*g*lc1*c1+m2*g*l1*c1;
    m2*g*lc2*c2];
%%%%%%%%%%%%%%%%%%%%%%%%%

%% Generating differential
[THDD]=inv(M)*(Tau_des- V*[th1d^2;th2d^2]-G); %%% Deriving angular acc
%% Simulation Update
dX(1,1)= th1d;
dX(2,1)= th2d;
dX(3,1)= THDD(1);
dX(4,1)= THDD(2);
%% Storage
itr=itr+1;
t7(itr)=t;
Q7(itr,1) = [th1];
Q7(itr,2) = [th2];
Q7(itr,3) = [th1d];
Q7(itr,4) = [th2d];
Q7(itr,5) = THDD(1);
Q7(itr,6) = THDD(2);
Tau4(itr,:)= Tau_des;
```

6) FIND_TAU_SIM.m

```
%%% CALCULATES TORQUE BASED ON JOINT SPACE INFORMATION FROM KINEMATIC
%%% SIMULATION

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function TAU_SIM=FIND_TAU_SIM(tspan,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8

for i=1:length(tspan)
    t=tspan(i);
    th1=X(i,1);
    th2=X(i,2);
%% Trajectory information
    R1=rx;
    R2=ry;
    r=ell_an;

    x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
    y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

    xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
    yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);

    xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
    ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);

    % th_des=invbot_new([x_des, y_des]); % position in joint space
    % th_des=invbot([x_des, y_des]); % position in joint space
    th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space

    %J=[-l1*sin(th_des(1)) -l2*sin(th_des(2));
    %  l1*cos(th_des(1))  l2*cos(th_des(2))];
    %% Simulation update
    J=[-l1*sin(th1) -l2*sin(th2);
        l1*cos(th1)  l2*cos(th2)];

    THD=inv(J)*[xd_des,yd_des]';
    dX=[THD];

    %% Determining angular accelerations
    th1d=THD(1);
    th2d=THD(2);

    Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
        -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];
%
%      inv(J)
%      ([xdd_des;ydd_des])
%       - Jdot*THD
    THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);
```

```
    %%%%%%%%%%%%%%%%%%%%%%% CREATING IMPORTANT MATRICES IN THE GOVERNING EQUATION
    s1=sin(th1);
    s2=sin(th2);
    c1=cos(th1);
    c2=cos(th2);
    c21=cos(th2-th1);
    s21=sin(th2-th1);

    % Kp=[1 0;
    %     0 1];
    %%%%%% Dynamic control matrices
    M = [ j1+m2*l1^2, m2*l1*lc2*c21;
        m2*l1*lc2*c21, j2+m2*lc2^2];
    V = [0, -m2*l1*lc2*s21;
        m2*l1*lc2*s21, 0];
    G = [m1*g*lc1*c1+m2*g*l1*c1;
        m2*g*lc2*c2];
    %%%%%%%%%%%%%%%%%%%%%%%%%
    TAU_SIM(i,:)= [M*[THDD(1);THDD(2)] + V*[th1d^2;th2d^2]+G]';
end
```

7) KINEMATIC ANALYSIS: OPEN LOOP

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [dX]=KIN_OL(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8

th1=X(1);
th2=X(2);

%% Trajectory information
R1=rx;
R2=ry;
r=ell_an;

x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);

xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);

% th_des=invbot_new([x_des, y_des]); % position in joint space
% th_des=invbot([x_des, y_des]); % position in joint space
th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space

%J=[-l1*sin(th_des(1)) -l2*sin(th_des(2));
  %  l1*cos(th_des(1))  l2*cos(th_des(2))];
%% Simulation update
J=[-l1*sin(th1) -l2*sin(th2);
    l1*cos(th1)  l2*cos(th2)];

THD=inv(J)*[xd_des,yd_des]';
dX=[THD];

%% Determining angular accelerations
th1d=THD(1);
th2d=THD(2);

Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
        -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];

THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);

%% Storage
itr=itr+1;
t1(itr)=t;
Q1(itr,1:2) = [th1, th2];
Q1(itr,3:4) = [th1d, th2d];
Q1(itr, 5:6) = [THDD]';
```

8) KINEMATIC ANALYSIS: CLOSED LOOP JOINT SPACE CONTROL

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [dX]=KIN_CLJS(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8

th1=X(1);
th2=X(2);

%% Trajectory information
R1=rx;
R2=ry;
r=ell_an;

x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);

xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);

% th_des=invbot_new([x_des, y_des]); % position in joint space
% th_des=invbot([x_des, y_des]); % position in joint space
th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space

% J=[-l1*sin(th_des(1)) -l2*sin(th_des(2));
%    l1*cos(th_des(1))  l2*cos(th_des(2))];
%% Simulation update
J=[-l1*sin(th1) -l2*sin(th2);
    l1*cos(th1)  l2*cos(th2)];
THD=inv(J)*[xd_des,yd_des]' + Kp*([th_des(1)-th1,th_des(2)-th2]');
dX=THD;

%% Determining angular accelerations
th1d=THD(1);
th2d=THD(2);

Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
       -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];

THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);


%% Storage
itr=itr+1;
t2(itr)=t;
Q2(itr,1:2) = [th1, th2];
Q2(itr,3:4) = [th1d, th2d];
Q2(itr, 5:6) = [THDD]';
```

9) KINEMATIC ANALYSIS CLOSED LOOP TASK SPACE CONTROL

```
% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [dX]=KIN_CLTS(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8

th1=X(1);
th2=X(2);

%% Trajectory information
R1=rx;
R2=ry;
r=ell_an;

x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);

xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);
% th_des=invbot_new([x_des, y_des]); % position in joint space
% th_des=invbot([x_des, y_des]); % position in joint space
th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space

%% Simulation update
J=[-l1*sin(th1) -l2*sin(th2);
    l1*cos(th1)  l2*cos(th2)];
x1= l1*cos(th1) + l2*cos(th2);
y1= l1*sin(th1) + l2*sin(th2);

THD=inv(J)*([xd_des,yd_des]' + Kx*([x_des-x1;y_des-y1]) );
dX=THD;

%% Determining angular accelerations
th1d=THD(1);
th2d=THD(2);

Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
        -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];

THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);

%% Storage
itr=itr+1;
t3(itr)=t;
Q3(itr,1:2) = [th1, th2];
Q3(itr,3:4) = [th1d, th2d];
Q3(itr, 5:6) = [THDD]';
```

10) INVBOT.m

```
%% RETURN JOINT SPACE ANGLES FOR A GIVEN END EFFECTOR POSITION

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah


% function to invert bot at time t and position x,y
% values always between -pi/3 and 2*pi/3
% risk of unfeasible space even with feasible solns when J is singular
% qdot = thdot and th1/th2 are the initial elbow up config. angles
function TH=invbot(X)
global l1 l2 %rx ry start_an ell_an w
l1=2; l2=1;
x=X(1);
y=X(2);
A=x^2+y^2+l1^2-l2^2+2*l1*x;
B=-4*l1*y;
C=x^2+y^2+l1^2-l2^2-2*l1*x;
th1=2*atan2(-B+sqrt(B^2-4*A*C),2*A); % using the eqn. (x-l1c1)^2+(y-l1s1)^2=l2^2
th2=atan2(y-l1*sin(th1),x-l1*cos(th1));
% xdot=-rx*sin(w*t+start_an)*cos(ell_an)*w-ry*cos(w*t+start_an)*sin(ell_an)*w;
% ydot=-rx*sin(w*t+start_an)*sin(ell_an)*w+ry*cos(w*t+start_an)*cos(ell_an)*w;
% J=[-l1*sin(th1) -l2*sin(th2);
%     l1*cos(th1)  l2*cos(th2)];
% qdot=inv(J)*[xdot;ydot]; % theta dot
TH=[th1 th2];% qdot'];
```

11) INVBOT2.m

```
%% RETURN JOINT SPACE ANGLES FOR A GIVEN END EFFECTOR POSITION AND AN
%% INITIAL START VALUE FOR JOINT SPACE ANGLES

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah


%% function invbot with 2pi additions till value is nearest to initial
%% approx. works best with all conditions. plan to use this.
% can give both q as th1/th2 and qdot as thetadot. for thd, include t as
% parameter.
function TH=invbot2(X,th)
global l1 l2%rx ry start_an ell_an w
l1=2; l2=1;
x=X(1);
y=X(2);
th1a=th(1);
th2a=th(2);
A=x^2+y^2+l1^2-l2^2+2*l1*x;
B=-4*l1*y;
C=x^2+y^2+l1^2-l2^2-2*l1*x;
th1=2*atan2(-B+sqrt(B^2-4*A*C),2*A)-2*pi;
th2=atan2(y-l1*sin(th1),x-l1*cos(th1))-2*pi;
prev=100;
while(abs(th1-th1a)<prev)
    prev=abs(th1-th1a);
    th1=th1+2*pi;
end
prev=100;
while(abs(th2-th2a)<prev)
    prev=abs(th2-th2a);
    th2=th2+2*pi;
end
% xdot=-rx*sin(w*t+start_an)*cos(ell_an)*w-ry*cos(w*t+start_an)*sin(ell_an)*w;
% ydot=-rx*sin(w*t+start_an)*sin(ell_an)*w+ry*cos(w*t+start_an)*cos(ell_an)*w;
% J=[-l1*sin(th1) -l2*sin(th2);
%     l1*cos(th1)  l2*cos(th2)];
% qdot=inv(J)*[xdot;ydot];
TH=[th1-2*pi th2-2*pi]';% qdot'];
```

12) TH_DES_INFO.m

```
%%%%%%%% EVALUATES DESIRED THETA, THETA_DOT, THETA_DOTDOT AT ANY GIVEN TIME

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function [THETA_DES]=TH_DES_INFO(t,X)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters


th1=X(1); %%%Initial approximation for theta 1
th2=X(2);

%% Trajectory information
R1=rx;
R2=ry;
r=ell_an;

x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);

xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);
% th_des=invbot_new([x_des, y_des]); % position in joint space
% th_des=invbot([x_des, y_des]); % position in joint space

th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space (using simulation
data for a better convergence)


J=[-l1*sin(th_des(1)) -l2*sin(th_des(2));
   l1*cos(th_des(1))  l2*cos(th_des(2))];

THD=inv(J)*([xd_des,yd_des]'); % Desired angular velocity


%% Determining angular accelerations
th1d=THD(1);
th2d=THD(2);

% Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
%         -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];
Jdot= [-l1*cos(th_des(1))*th1d, -l2*cos(th_des(2))*th2d;
        -l1*sin(th_des(1))*th1d, -l2*sin(th_des(2))*th2d;];

THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);    % Desired angular accelerations

% THETA_DES=[th1,th2,THD',THDD'];
THETA_DES=[th_des(1),th_des(2),THD',THDD'];
```

13) PLOTBOT_JS.m

```
%%% FUNCTION FOR PLOTTING THE SIMULATED OUTPUT FOR A GIVEN TIME VECTOR AND
%%% JOINT SPACE ANGLE VECTORS

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function plotbot_js(t,X,index,txt1)
global l1 l2 rx ry ell_an start_an w A B
aviobj = avifile([txt1,'.avi'],'compression','Cinepak'); % Declare an avi object

h=figure(index*3-2);
cla('reset');
axis manual;
axis([-3 3 -3 3]);
hold on;
grid on;
c_ell_an=cos(ell_an);
s_ell_an=sin(ell_an);
plot(A+rx*cos(w*t)*c_ell_an-ry*sin(w*t)*s_ell_an,...
     B+rx*cos(w*t)*s_ell_an+ry*sin(w*t)*c_ell_an,'-k');
title(txt1);
for i=1:length(t)
    th1=X(i,1);
    th2=X(i,2);
    x1=l1*cos(th1);
    y1=l1*sin(th1);
    x2=x1+l2*cos(th2);
    y2=y1+l2*sin(th2);
%     plot([0 x1 x2],[0 y1 y2]);
    plot(x2,y2,'ro');
%     pause(0.03);
    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;
```

14) ERROR_PLOT_1.m

```
%%%% ERROR CALCULATION FOR KINEMATIC ANALYSIS

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah


function []= ERROR_PLOT_1(tspan,X,h,txt1)


global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8


for i=1:length(tspan)
    t=tspan(i);
    th1=X(i,1);
    th2=X(i,2);


    %% Trajectory information
    R1=rx;
    R2=ry;
    r=ell_an;

    x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
    y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

    xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
    yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);

    xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
    ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);

    % th_des=invbot_new([x_des, y_des]); % position in joint space
    % th_des=invbot([x_des, y_des]); % position in joint space
    th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space

    %J=[-l1*sin(th_des(1)) -l2*sin(th_des(2));
    %  l1*cos(th_des(1))  l2*cos(th_des(2))];
    %% Simulation update
    J=[-l1*sin(th1) -l2*sin(th2);
        l1*cos(th1)  l2*cos(th2)];

    THD=inv(J)*[xd_des,yd_des]';
    dX=[THD];

    %% Determining angular accelerations
    th1d=THD(1);
    th2d=THD(2);

    Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
        -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];
%
%      inv(J)
```

```
%       ([xdd_des;ydd_des])
%        - Jdot*THD
    THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);

    xa= l1*cos(th1) + l2*cos(th2); %%% Actual end effector positions (TS)
    ya= l1*sin(th1) + l2*sin(th2);
    x_err(i,:)= [x_des - xa, y_des - ya, sqrt((x_des - xa)^2+(y_des - ya)^2)]; %%% Task
Space Error
    th_err(i,:)= [th_des(1) - th1, th_des(2) - th2, sqrt((th_des(1) - th1)^2+(th_des(2) -
th2)^2)];%%% Joint Space Error
end
% Error plot
figure(3*h-1)
subplot(3,1,1)
plot(tspan,x_err(:,1));
ylabel('xd-x');
title([txt1,':End Effector position error plot']);
subplot(3,1,2)
plot(tspan,x_err(:,2));
ylabel('yd-y')
subplot(3,1,3)
plot(tspan,x_err(:,3));
ylabel('norm(Xd-X)')
xlabel('Time (sec)');

figure(3*h)
subplot(3,1,1)
plot(tspan,th_err(:,1));
ylabel('th1d-th1');
title([txt1,':Joint Space error plots']);
subplot(3,1,2)
plot(tspan,th_err(:,2));
ylabel('th2d-th2')
subplot(3,1,3)
plot(tspan,th_err(:,3));
ylabel('norm(THd-TH)')
xlabel('Time (sec)');
```

15) ERROR_PLOT_2.m

```
%%%% ERROR CALCULATION FOR DYNAMIC ANALYSIS

% Course: Robotic Manipulation and Mobility
% Advisor: Dr. V. Krovi
%
% Homework Number: 4
%
% Names: Sourish Chakravarty
%     Hrishi Lalit Shah

function []= ERROR_PLOT_2(tspan,X,h,txt1)

global l1 l2 lc1 lc2 j1 j2 m1 m2 g rx ry ell_an w Kp Kx Kp1 Kd1 A B % Given parameters
global itr Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 t1 t2 t3 t4 t5 t6 t7 t8


for i=1:length(tspan)
    t=tspan(i);
    th1=X(i,1);
    th2=X(i,2);
    th1d=X(i,3);
    th2d=X(i,4);

    %% Trajectory information
    R1=rx;
    R2=ry;
    r=ell_an;

    x_des= A + R1*cos(w*t)*cos(r)-R2*sin(w*t)*sin(r);
    y_des= B + R1*cos(w*t)*sin(r)+R2*sin(w*t)*cos(r);

%     xd_des= -R1*w*sin(w*t)*cos(r)-R2*w*cos(w*t)*sin(r);
%     yd_des= -R1*w*sin(w*t)*sin(r)+R2*w*cos(w*t)*cos(r);
%
%     xdd_des = -R1*(w^2)*cos(w*t)*cos(r)+R2*(w^2)*sin(w*t)*sin(r);
%     ydd_des = -R1*(w^2)*cos(w*t)*sin(r)-R2*(w^2)*sin(w*t)*cos(r);

    % th_des=invbot_new([x_des, y_des]); % position in joint space
    % th_des=invbot([x_des, y_des]); % position in joint space
    th_des=invbot2([x_des, y_des],[th1, th2]); % position in joint space

    %J=[-l1*sin(th_des(1)) -l2*sin(th_des(2));
    %  l1*cos(th_des(1))  l2*cos(th_des(2))];
    %% Simulation update
%     J=[-l1*sin(th1) -l2*sin(th2);
%         l1*cos(th1)  l2*cos(th2)];
%
%     THD=inv(J)*[xd_des,yd_des]';
%     dX=[THD];

    %% Determining angular accelerations
%     th1d=THD(1);
%     th2d=THD(2);
%
%     Jdot= [-l1*cos(th1)*th1d, -l2*cos(th2)*th2d;
%         -l1*sin(th1)*th1d, -l2*sin(th2)*th2d;];
%
%      inv(J)
%      ([xdd_des;ydd_des])
```

```
%       - Jdot*THD
%     THDD= inv(J)*([xdd_des;ydd_des] - Jdot*THD);

    xa= l1*cos(th1) + l2*cos(th2); %%% Actual end effector positions (TS)
    ya= l1*sin(th1) + l2*sin(th2);
    x_err(i,:)= [x_des - xa, y_des - ya, sqrt((x_des - xa)^2+(y_des - ya)^2)];
    th_err(i,:)= [th_des(1) - th1, th_des(2) - th2, sqrt((th_des(1) - th1)^2+(th_des(2) -
th2)^2)];
end
figure(3*h-1)
subplot(3,1,1)
plot(tspan,x_err(:,1));
ylabel('xd-x');
title([txt1,':End Effector position error plot']);
subplot(3,1,2)
plot(tspan,x_err(:,2));
ylabel('yd-y')
subplot(3,1,3)
plot(tspan,x_err(:,3));
ylabel('norm(Xd-X)')
xlabel('Time (sec)');

figure(3*h)
subplot(3,1,1)
plot(tspan,th_err(:,1));
ylabel('th1d-th1');
title([txt1,':Joint Space error plots']);
subplot(3,1,2)
plot(tspan,th_err(:,2));
ylabel('th2d-th2')
subplot(3,1,3)
plot(tspan,th_err(:,3));
ylabel('norm(THd-TH)')
xlabel('Time (sec)');
```