Two-Link Manipulator

Mini Project - Systems Thinking (EC5.202)
Team - **Think2Link**

Abhinav S. - 2022102037
Aryaman Mahajan - 2022102034
Harshvardhan Singh - 2022112004
Koustubh Jain - 2023122003
Om Mehta - 2022102046
Rupak Antani - 2022102045
Soham Vaishnav - 2022112002
Soumil Gupta - 2022102035

October 7, 2023

Contents

1 Introduction

The given system is of a 2-link robotic arm. The system is like a double pendulum system with a movable joint connecting the two rods while a hinged joint connects one rod to the floor. As given in the problem statement, we must design controllers for the system such that the system, in the initial [0.1 rad 0.1 rad] configuration comes to the [0 rad 0 rad] configuration. We propose a proportional integral derivative controller which does the required task in an efficient manner, while ensuring that the steady state error is minimised. There is an application of 2 torques to achieve the desired result.

2 Two-Link Manipulator

A 2 link-manipulator is a mechanical system which has 2 degrees of freedom consisting of two rigid links connected via joints. The joints allow the robot to move via 2 independent rotational degrees of freedom. The key components are:

2.1 Links

Links are rigid segments of the manipulator The system consists of 2 Links namely q_1 and q_2, q_1 is connected to a fixed point/base while q_2 is attached to q_1.

2.2 Joints

These are the connections between the links, one of them is fixed at the base while the other is free to move with link q_1, they allow relative rotational motion about them for the links.

2.3 Workspace

This refers to the region that the system can reach. For this specific system it refers to the area that the end of q_2 link (which is the effect arm) covers.

2.4 Kinematics and Inverse Kinematics

This is the most relevant component of our project, it refers to the study of determining what joint angles we need to position the end effector (q_2) at a certain location/orientation ((0,0) in this case) and what forces (Torque) are needed to achieve that task, this involves a mathematical description which quantifies the forces and relates them to the velocity, acceleration, location etc- of the links via state space equations. This is an essential component in robotics and control design.

3 System Dynamics of a Two-link Manipulator

As shown in the figure below, the 2-link manipulator consists of 2 masses, m_1 and m_2 connected by weightless bars of length l_1 and l_2. Also q_1 is the angle rotated by the first bar about the origin and q_2 is the angle rotated by the second bar about the joint of the two bars.

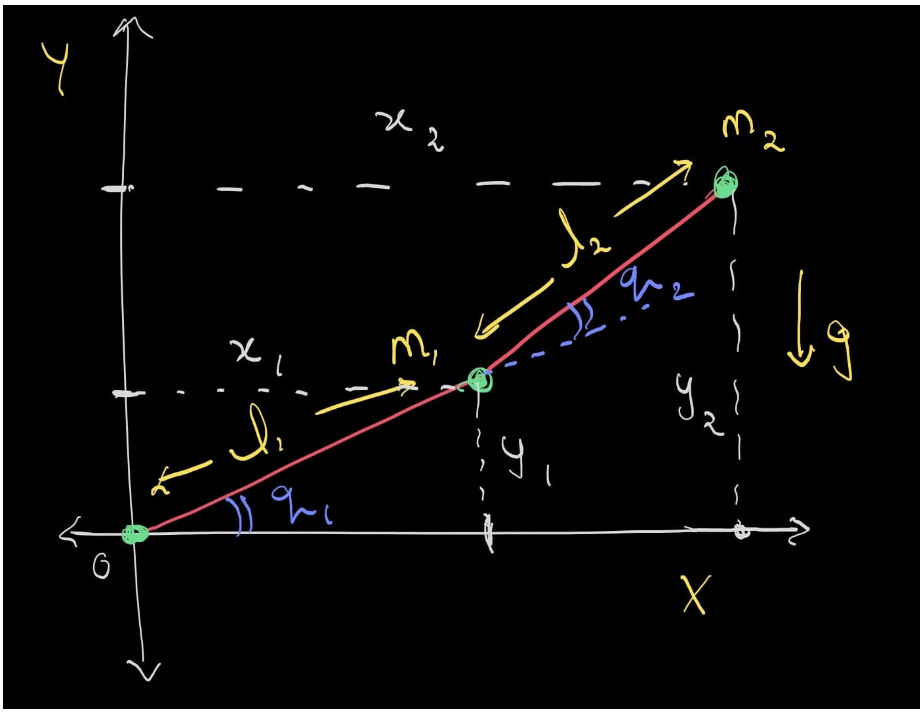

Figure 1: Two-Link Manipulator

Let x_1 and y_1 be the co-ordinates of m_1 and let x_2 and y_2 be the co-ordinates of m_2. Now,

$$x_1 = l_1 \cos(q_1) \tag{1}$$
$$y_1 = l_1 \sin(q_1) \tag{2}$$
$$x_2 = l_1 \cos(q_1) + l_2 \cos(q_2) \tag{3}$$
$$y_2 = l_1 \sin(q_1) + l_2 \sin(q_2) \tag{4}$$
$$\tag{5}$$

Therefore, the velocities of the two masses will be:

$$v_1 = \sqrt{(\dot{x_1})^2 + (\dot{y_1})^2} \tag{6}$$
$$v_2 = \sqrt{(\dot{x_2})^2 + (\dot{y_2})^2} \tag{7}$$
$$\tag{8}$$

where,

$$\dot{x_1} = -l_1\dot{q_1}\sin q_1 \tag{9}$$
$$\dot{y_1} = l_1\dot{q_1}\cos q_1 \tag{10}$$
$$\dot{x_2} = -l_1\dot{q_1}\sin q_1 - l_2\dot{q_2}\sin q_2 \tag{11}$$
$$\dot{y_2} = l_1\dot{q_1}\cos q_1 + l_2\dot{q_2}\cos q_2 \tag{12}$$
$$\tag{13}$$

Here, $\dot{}$ represents derivative with respect to time.

Now, the total kinetic energy can be gven by:

$$KE = \frac{m_1v_1^2}{2} + \frac{m_2v_2^2}{2}$$

$$KE = \frac{1}{2}m_1(\dot{x_1}^2 + \dot{y_1}^2) + \frac{1}{2}m_2(\dot{x_2}^2 + \dot{y_2}^2)$$

$$KE = \frac{1}{2}m_1((-l_1\dot{q_1}\sin q_1)^2 + (l_1\dot{q_1}\cos q_1)^2) + \frac{1}{2}m_2((-l_1\dot{q_1}\sin q_1 - l_2\dot{q_2}\sin q_2)^2 + (l_1\dot{q_1}\cos q_1 + l_2\dot{q_2}\cos q_2)^2)$$

$$KE = \frac{1}{2}(m_1+m_2)l_1^2\dot{q_1}^2 + \frac{1}{2}(m_2)l_2^2\dot{q_2}^2 + \frac{1}{2}m_2l_1l_2\dot{q_1}\dot{q_2}\cos(q_1-q_2)$$

Also, the total potential energy of the system can be given by:

$$PE = \sum_{i=1}^{2} PE_i(q) = \sum_{i=1}^{2} m_igh_i(q)$$

$$PE = (m_1+m_2)gl_1\sin q_1 + m_2gl_2\sin q_2$$

where h_i is the height of the center of mass, g is the acceleration due to gravity and m_i is the mass of the ith pendulum.

Next by Lagrange Dynamics,the Lagrangian $\mathcal{L}$ is defined as

$$\mathcal{L} = KE - PE$$

Substituting the values for kinetic energy and potential energy previoulsly obtained,

$$\mathcal{L} = \frac{1}{2}(m_1+m_2)l_1^2\dot{q_1}^2 + \frac{1}{2}(m_2)l_2^2\dot{q_2}^2 + m_2l_1l_2\dot{q_1}\dot{q_2}\cos(q_1-q_2) - (m_1+m_2)gl_1\sin q_1 - m_2gl_2\sin q_2$$

Therefore, using the Euler-Lagrange Equation we obtain the following two non-linear equations of motion which are second order ordinary differential equations

$$l_1\ddot{q_1} + \delta l_1l_2\ddot{q_2}\cos q_1 - q_2 = \frac{\delta\tau_1}{m_2l_1} - \delta l_2\dot{q_2}^2\sin(q_1-q_2) - g\cos q_1$$

$$l_1\ddot{q}_2 + l_1\ddot{q}_1 \cos(q_1 - q_2) = \frac{\tau}{m_2 l_2} + l_1 \dot{q}_1{}^2 \sin(q_1 - q_2) - g\cos q_2$$

where, $\delta = \dfrac{m_2}{m_1 + m_2}$

From the above equations, we get the following values for $\ddot{q}_1$ and $\ddot{q}_2$

$$\ddot{q}_1 = g_1(t, q_1, q_2, \dot{q}_1, \dot{q}_2), \ddot{q}_2 = g_2(t, q_1, q_2, \dot{q}_1, \dot{q}_2),$$

where,

$$g_1 = \frac{\frac{\delta\tau_1}{m_2 l_1} - \delta l_2 \dot{q}_2^2 \sin(q_1 - q_2) - g\cos(q_1) - \delta\cos(q_1 - q_2)\left(\frac{\tau_2}{m_2 l_2} + l_1\dot{q}_1^2 \sin(q_1 - q_2) - g\cos(q_2)\right)}{l_1\left(1 - \delta\cos^2(q_1 - q_2)\right)}$$

$$g_2 = \frac{\frac{\tau_2}{m_2 l_2} + l_1 \dot{q}_1^2 \sin(q_1 - q_2) - g\cos(q_2) - \cos(q_1 - q_2)\left(\frac{\delta\tau_1}{m_2 l_1} - \delta l_2\dot{q}_2^2 \sin(q_1 - q_2) - g\cos(q_1)\right)}{l_2\left(1 - \delta\cos^2(q_1 - q_2)\right)}$$

In order to solve for the angles q_1 and q_2, let us consider four variables

$$u_1 = q_1, u_2 = q_2, u_3 = \dot{q}_1, u_4 = \dot{q}_2.$$

After differentiating with respect to time,

$$\dot{u}_1 = \dot{q}_1 = u_3, \dot{u}_2 = \dot{q}_2 = u_4, \dot{u}_3 = \ddot{q}_1 = g_1(t, u_1, u_2, u_3, u_4), \dot{u}_4 = \ddot{q}_2 = g_2(t, u_1, u_2, u_3, u_4)$$

Thus, we obtain a system of first-order nonlinear differential equations of the form

$$\frac{dU}{dt} = S(t, U), U(0) = U_0,$$

where $U = [u_1, u_2, u_3, u_4]^t$ and $S = [s_1, s_2, s_3, s_4]^t$ with

$$s_1 = u_3, s_2 = u_4, s_3 = g_1(t, u_1, u_2, u_3, u_4), s_4 = g_2(t, u_1, u_2, u_3, u_4)$$

The initial conditions are given by

$$U_0 = [u_1(0), u_2(0), u_3(0), u_4(0)]^t$$

Where,

$$u_1(0) = q_1(0), u_2(0) = q_2(0), \quad u_3(0) = \dot{q}_1(0), u_4(0) = \dot{q}_2(0)$$

4 State Space Model

The state-space model is a way of representing the system such that it becomes easier to compute and analyse the time domain responses involved in system dynamics. In order to achieve this, the state-space model involves a set of equations which are first order ODEs that make computation easier.
As discussed while deriving the system dynamics, we have already defined the four primary states as -

$$x_1 = q_1(t),\ x_2 = q_2(t),\ x_3 = \dot{q}_1(t),\ x_4 = \dot{q}_2(t)$$

We also have the system dynamics equation as -

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

Now let the states matrices be noted as $\dot{X}$ and X, where

$$\dot{X} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} \text{ and } X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

and input as we already know is τ which is also a column matrix as shown below:

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

On closer observation of the system dynamics equation, we recognise that the term multiplied to matrix $M(q)$ is a second order differential matrix of q, which if looked from the perspective of the state-space model, denotes the a part of the states matrix $\dot{X}$.
Therefore rearranging the terms of the system dynamics equation, we get -

$$\dot{S_1} = -M(q)^{-1}(C(S, S_1)S_1 + G(S) - \tau)$$

where

$$S = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } S_1 = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}.$$

The above equation essentially represents the state-space model required to analyse the two-link system.
The forcing function matrix (τ) is where the controller dynamics will be incorporated. The input is where the controls are applied and the design of these controls is explained in the following section.

5 Controller Design

Before describing the controller dynamics and parameters, it is better to get an overview of what controllers might suit the system.

5.1 P control

The two-link system when left unattended follows a chaotic path and in cases of an ideal two-link pendulum, the system never stabilises because the moment it reaches steady state, the internal forces of the system drive it to an unstable configuration.
Use of only **proportional** control guarantees that the steady state error will decrease, but not necessary will it reach the final desired steady state (which is 0 in our case). Another aspect of proportional control is that it will make the transient response time efficient, thereby making the response faster.
This is because the proportional constant K_p contributes to the ω_n^2 term in the denominator of the standard second-order closed loop transfer function which then becomes -

$$G(s) = \frac{K_p\omega_n^2}{s^2+2s^2\zeta\omega_n+(\omega_n^2+K_p)}.$$

It is implemented by placing the proportional gain block in series with and before the open loop system transfer function $H(s)$. Therefore, by choosing sufficiently large value of K_p we can essentially speeden up the system's transient response and reduce steady-state error.

5.2 PD control

Here, the control is includes a derivative in terms of the error alongside proportional gain. Therefore, the controller now looks as -

$$K_p + K_d s$$

where K_d is the derivative control. This controller is implemented by placing the derivative block parallel to the proportional gain block. Clearly, the derivative term contributes to the decaying nature of the transient response and in reducing the oscillatory nature of the transient response given some inherent value of ζ that the system has. That is, if ζ is much smaller, then it may give rise to complex roots which introduce some unavoidable oscillations. But, by adding K_d to it, we can reduce the oscillitions the system takes to reach the steady state which in turn is determined by K_p as discussed earlier.
However, we need to be mindful of the fact that the system may or may not reach the desired steady state.

5.3 PI control

Here, the control includes the proportional gain term alongside an error-integral term and is built by adding an integral block in parallel to the proportional gain block. The control therefore looks as -

$$K_p + K_i/s.$$

An interesting point to note is that this control design makes the system third order by introducing a third root. However, a general analysis suggests that the transient response tends to reach closer to set state due to the integral term and its rate of reaching there is determined by the proportional gain. Generally, overall response of a PI controller is better than that of the PD controller but in terms of settling time, PD is better as it lessens the oscillations and fastens the response.

5.4 PID Control

As the name suggests, this controller included all the controls - proportional gain, derivative and integral - and therefore it is an interplay of the constants K_p, K_d, K_i that determine how the system will respond.
This controller design takes care of the response time (P and D), oscillations (I and D) as well as the attainment of set state (P and I).
Under this configuration, the closed loop transfer function of the system looks as follows -

$$G(s) = \frac{(K_p + K_d s + K_i/s)\omega_n^2}{s^3 + 2s^2(\zeta\omega_n + K_d) + (\omega_n^2 + K_p)s + K_i}.$$

This controller is implemented by placing all the three control blocks in parallel with each other such that the entire setting is in series and before the open loop system transfer function.

5.5 Controller design for Two-link Manipulator

Here, we chose to apply the PID control, and depending on the responses, we tried and adjusted the values of K_p, K_d and K_i. As discussed earlier, the controller parameters are applied to the inputs to the system which in our case is the matrix τ.
Let the error be denoted as e. Now, in our case, the intial conditions and final conditions are as follows -

$$S_i = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}, S_{1_i} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \dot{S_{1_i}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and

$$S_f = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

where subscripts i and f stand for intial and final conditions respectively.
Therefore, the error term e can be written as difference between set (or final or desired) state and the state of the system at any point in time denoted by matrices S and S_1. Therefore,

$$e = -S, \text{ and } \dot{e} = -\dot{S} = -S_1.$$

Applying controls to the input matrix τ using the constants K_p, K_d, and K_i, we get -

$$\tau = M(q)(K_p e + K_d \dot{e} + K_i \textstyle\int e)$$
$$\therefore \tau = M(q)(K_p(-S) + K_d(\dot{-S}) + K_i \textstyle\int(-S)),$$

further breaking it down into two torques, we get

$$\tau_1 = M(q)(K_p(-x_1) + K_d(\dot{-x_1}) + K_i \textstyle\int(-x_1)),$$
$$\tau_2 = M(q)(K_p(-x_2) + K_d(\dot{-x_2}) + K_i \textstyle\int(-x_2))$$

As we can observe, there are additional two terms -

$$\textstyle\int e_1, \text{ and } \int e_2.$$

If we try to bring these terms to their zero order form, then we will have to incorporate third degree differentials in terms of q_1 and q_2. Therefore, for simplicity, let these two terms define two new states x_5 and x_6.
The total number of states now becomes 6 - 4 as preciously defined and 2 being the integral ones.
Let the matrix defining these two new states be depicted as -

$$S_2 = \begin{bmatrix} x_5 \\ x_6 \end{bmatrix}$$

Let us now revisit the state-space model and following changes are made to it -

$$x_1 = q_1, x_2 = q_2, x_3 = \dot{q_1}, x_4 = \dot{q_2}, x_5 = \textstyle\int e_1, x_6 = \int e_2,$$

thus,

$$\dot{S_1} =$$
$$-M(q)^{-1}(C(S, S_1)S_1 + G(S) - \begin{bmatrix} M(q)(K_p(-x_1) + K_d(\dot{-x_1}) + K_i \int(-x_1)) \\ M(q)(K_p(-x_2) + K_d(\dot{-x_2}) + K_i \int(-x_2)) \end{bmatrix})$$

which now becomes,

$$\dot{S_1} = -M(q)^{-1}(C(S, S_1)S_1 + G(S) - M(q)(K_p(-S) + K_d(-S_1) + K_i(-S_2)))$$

, where S, S_1 and S_2 are as defined earlier.
On further simplification, we can get two seperate equations for $\ddot{q_1}$ and $\ddot{q_2}$ but it may increase the complexity in terms of understanding what is happening in the system. However, one possible simplification results in the following equation -

$$\dot{S_1} = -M(q)^{-1}(C(S, S_1)S_1 + G(S)) + (K_p(S) + K_d(S_1) + K_i(S_2)).$$

This, now is the final state-space model that can be used to code and understand the system performance, by varying the values of K_p, K_d and K_i to see difference in control designs.
The same has been discussed in the following section.

6 Simulations using MATLAB

6.1 Matlab Code

The equations controlling the system dynamics of a two-link manipulator can be solved using a special inbuilt function known as ode45. The ode45 function is MATLAB's standard solver for ordinary differential equations (ODEs). ode45 is designed to handle the following general problem:

$$\frac{dx}{dt} = f(t, x), \quad x(t_0) = x_0$$

where t is the independent variable, x is a vector of dependent variables to be found and f(t, x) is a function of t and x. The below code snippet shows the matrices M,C and G and the ode45 function:

```
zinit = [0, 0.1,0, 0, 0.1, 0];
tspan = [0, 20];

M = @(q2)[(m1 + m2)*l1^2 + m2*l2*(l2 + 2*l1*cos(q2)),
    m2*l2*(l2 + l1*cos(q2));
                m2*l2*(l2 + l1*cos(q2)), m2*(l2^2)];

C = @(q, qdot)[-m2*l1*l2*sin(q(2))*qdot(2), -m2*l1*l2*
    sin(q(2))*(qdot(1) + qdot(2));
                    0, m2*l1*l2*sin(q(2))* qdot(2)];

G = @(q)[(m1*l1*g*cos(q(1))) + (m2*g*((l2*cos(q(1) + q
    (2))) + (l1*cos(q(1)))));
              m2*g*l2*cos(q(1) + q(2))];


[T, x] = ode45(@(t, z) statespace(t, z, M_fun, C_fun,
    G_fun), tspan, zinit);
```

Here, the zinit variable gives the initial conditions of all the states involved and tspan is the time span for which we are plotting the graph of q_1 and q_2.The below code snippet shows the 'statespace' function:

```
function y = statespace(~, z, M_fun, C_fun, G_fun)

    y = zeros(size(z));
    qdot = [z(3);z(6)];
    q = [z(2);z(5)];
    ie = [z(1);z(4)];
    q2 = z(5);
    M = M_fun(q2);
    C = C_fun(q, qdot);
    G = G_fun(q);
    q1_d = 0;     q2_d = 0;     q_d = [q1_d ; q2_d];
    Kp = [2;2];     Kd = [2;2];      Ki = [2;2];

    M_inverse = inv(M);
    Y = -M_inverse * ((C * qdot) + G) +(Kp .*(q_d-q))
        -(Kd .* qdot) +(Ki .* ie);

    y(1)=q1_d-z(2);      y(2)=z(3);      y(3)=Y(1);
    y(4)=q2_d-z(5);      y(5)=z(6);      y(6)=Y(2);
end
```

The code starts by using the initial conditions provided to ode45. The statespace function calculates the expression for derivatives of all the states which are provided in the function as a function of the states and the input,the desired angle and gives the matrix y with value of the derivatives. This function is executed again and again to get the value of the states across the time interval provided in the timespan.

6.2 Simulink

In Simulink, the same system dynamics has been implemented using blocks. The reference provides the desired final angles. The controller calculates the output value using the constants and the error provided to it. This along with our states is provided to the state space subsystem which calculates the second order derivative term which is just a function of the the angles, their first derivatives and the output of the controllers as mentioned in the system dynamics. This is integrated to get the first order derivative terms and then again to get the angles. The initial condition of the angles is provided inside the integrator block. The angle is used to calculate the error along with reference using a subtractor block. Now,we have the next set of value of the states. So,the simulation starts with the initial condition provided to it by the integrator block. Calculates the next values of the states using state space subsystem block and integrator blocks. This continues on the timespan for which the simulation is run.

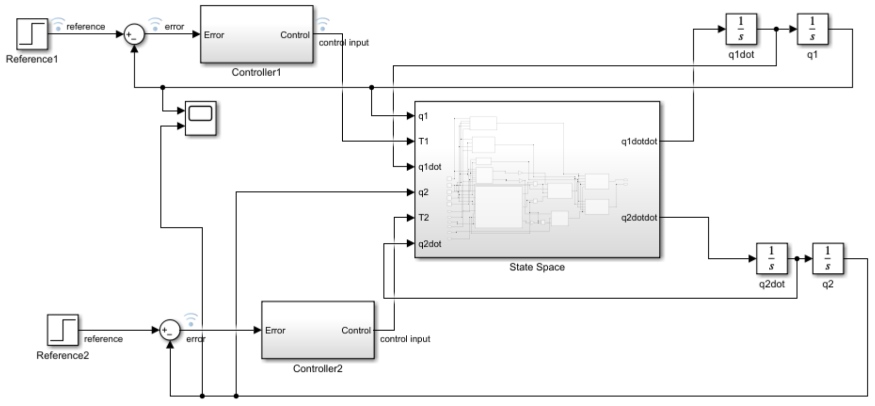

Figure 2: Simulink

7 Inference

7.1 No Controller

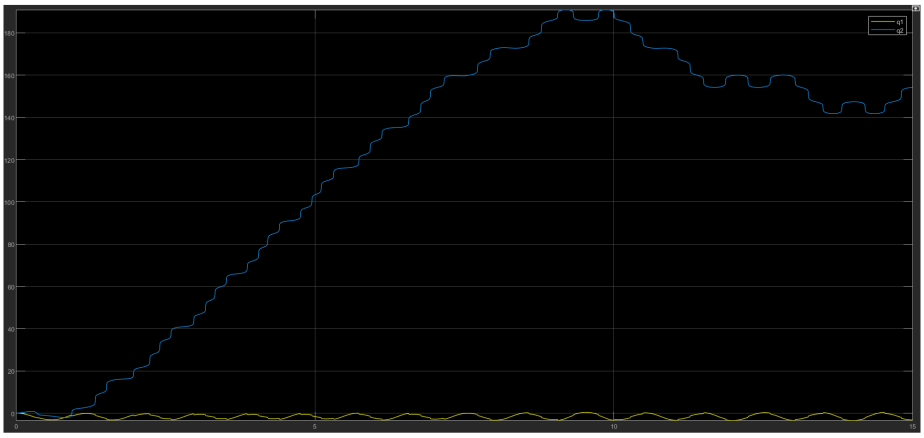

Figure 3: No Controller

Without any controller, the system does reach the desired state at some point, but becomes highly unstable after that.

7.2 P Control

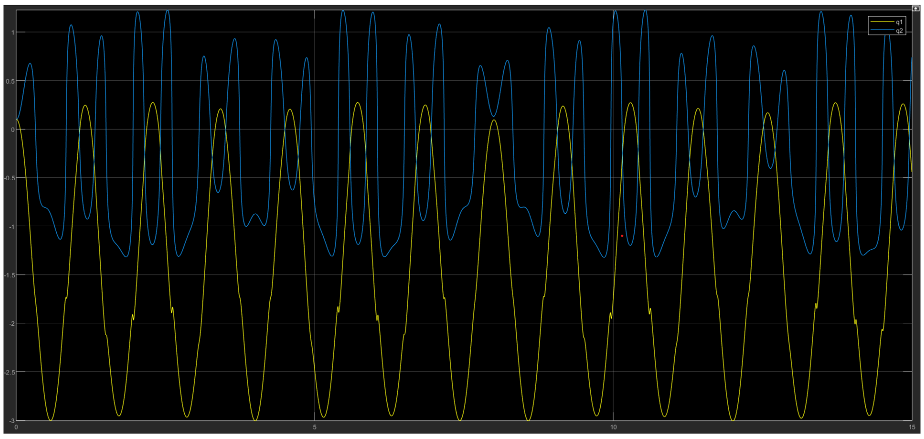

Figure 4: K_{p_1}=0.1,K_{p_2}=0.1

We are getting a sluggish response reacting very slowly. Depending on the system dynamics it might not even achieve the desired value at all. The system is unstable and does not reach a steady state at all.

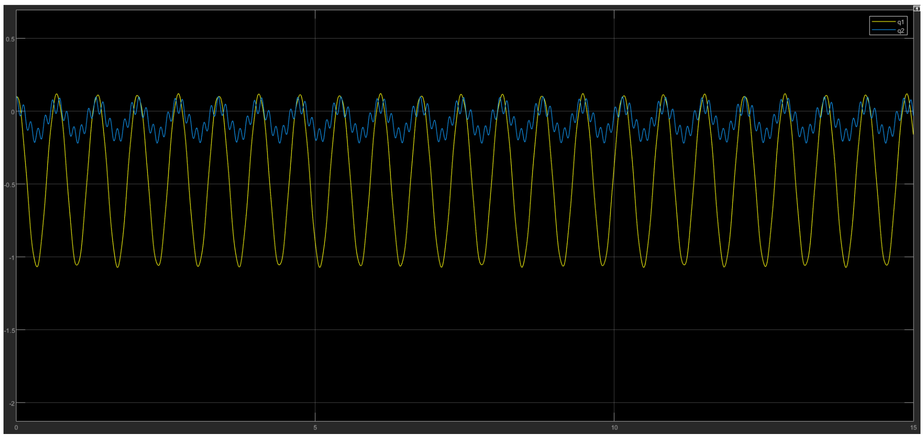

Figure 5: K_{p_1}=60,K_{p_2}=60

Increasing P, we see a faster response towards the set value though it never stablizes to a value. We also observe a decrease in steady state error.

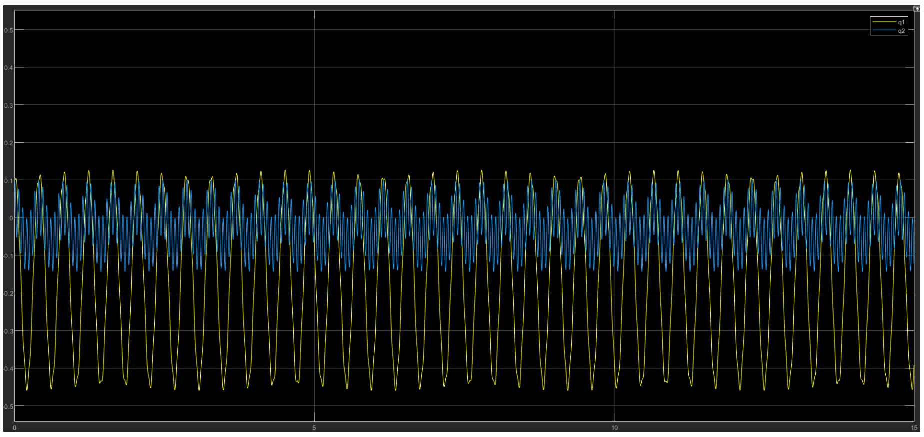

Figure 6: K_{p_1}=200,K_{p_2}=200

Increasing P further is causing the system to vibrate rapidly which is not a good result.

7.3 PD Control

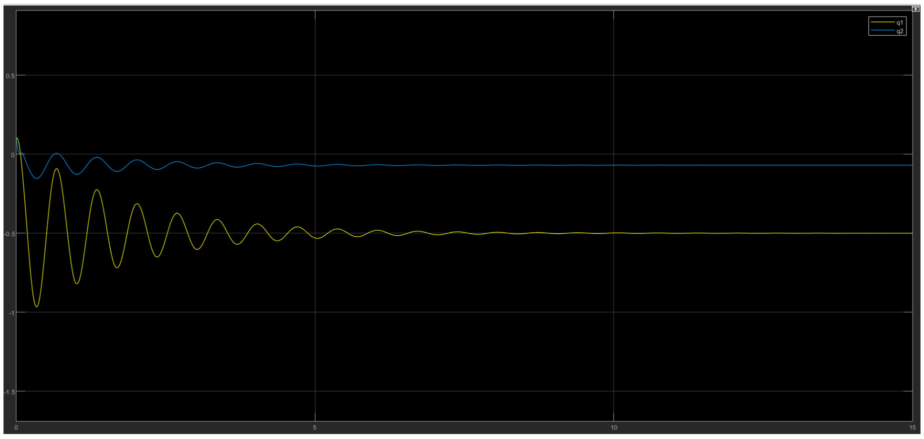

Figure 7: K_{p_1}=60,K_{p_2}=60,K_{d_1}=1,K_{d_2}=1

Here, K_p is appropriately set. Introducing a derivative term reduces the oscillations in the transient response and reduces the change in angles when error is changing rapidly. It also causes the system to stabilise though the steady state error is not zero.

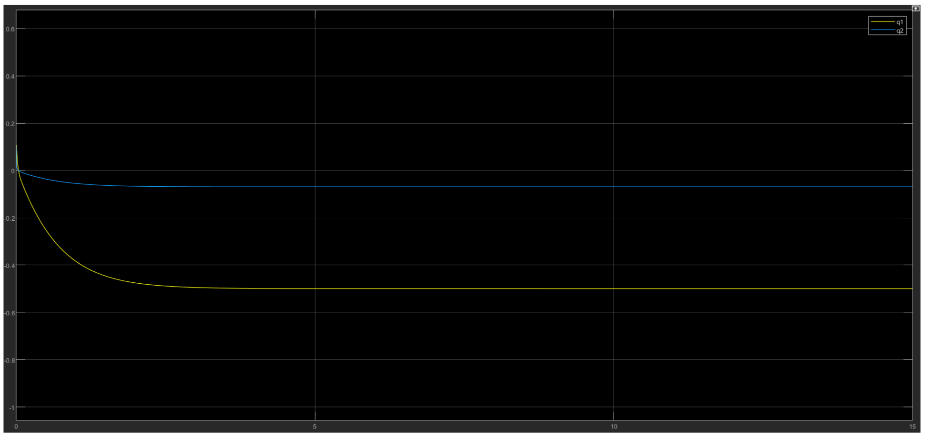

Figure 8: K_{p_1}=60,K_{p_2}=60,K_{d_1}=50,K_{d_2}=50

Increasing K_d keeping K_p constant, we observe that the oscillation decreases further to a point we are almost getting a smooth curve. It stabilizes to a steady value which is same as that in the previous case.

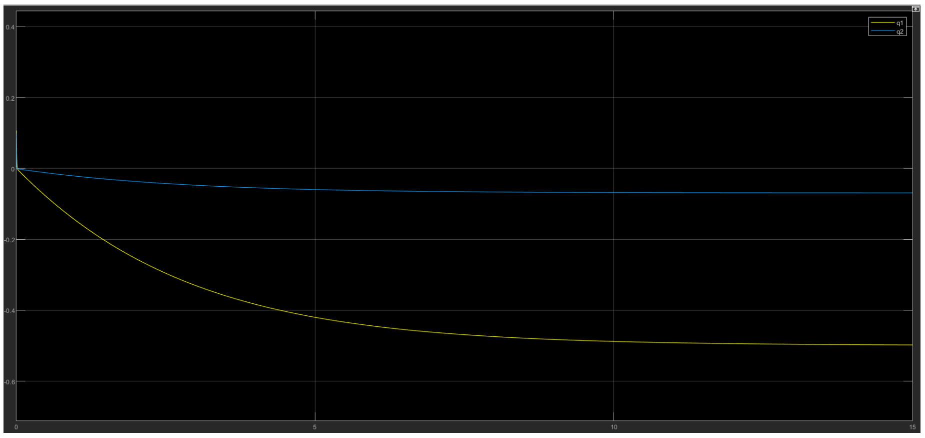

Figure 9: K_{p_1}=60,K_{p_2}=60,K_{d_1}=200,K_{d_2}=200

Increasing K_d further, we observe that the response becomes sluggish. Very high derivative term kind of resists change in the angle though the steady state is still achieved.

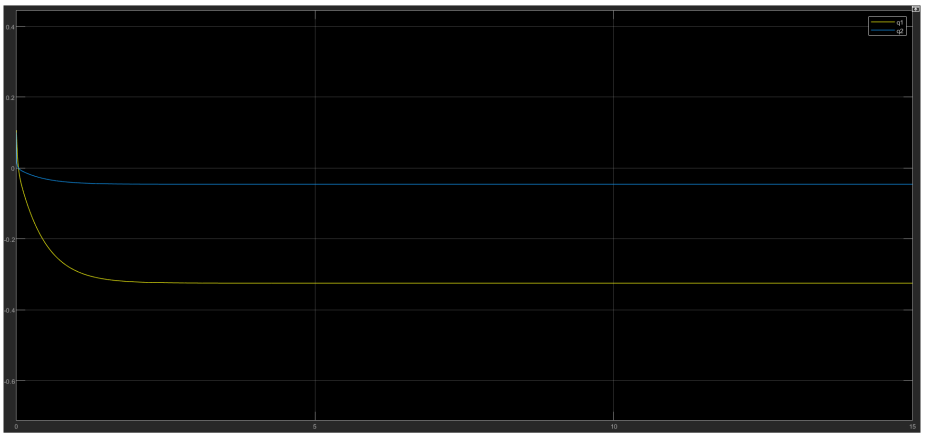

Figure 10: K_{p_1}=100,K_{p_2}=100,K_{d_1}=50,K_{d_2}=50

Increasing K_p keeping K_d constant, we observe a faster transient response and a decrease in steady state error.

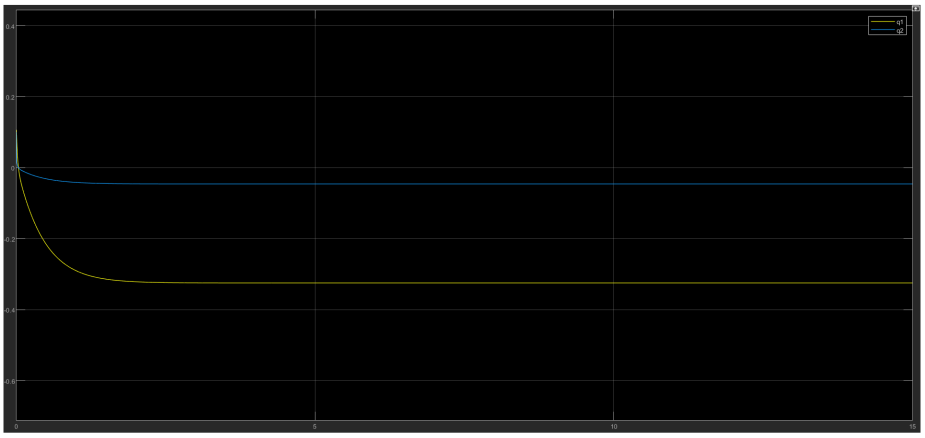

Figure 11: K_{p_1}=200,K_{p_2}=200,K_{d_1}=50,K_{d_2}=50

K_p when set high in P controller, we observed a rapid vibration of the system. Introducing a derivative term limits the rapid rotation of the system. Also, we see a further decrease in steady state error.

7.4 PI Control

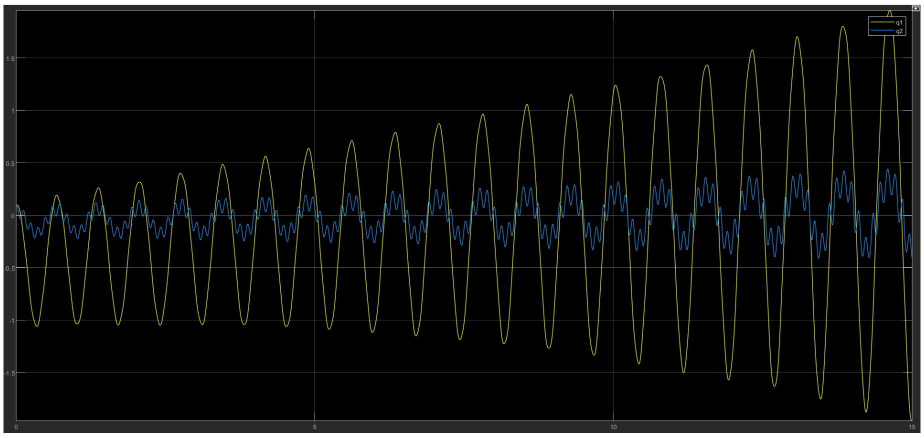

Figure 12: K_{p_1}=60,K_{p_2}=60,K_{i_1}=10,K_{i_2}=10

Here, we observe that both the outputs achieve the desired state but the output becomes highly unstable and the angles between which the oscillation happens is keeps on increasing.

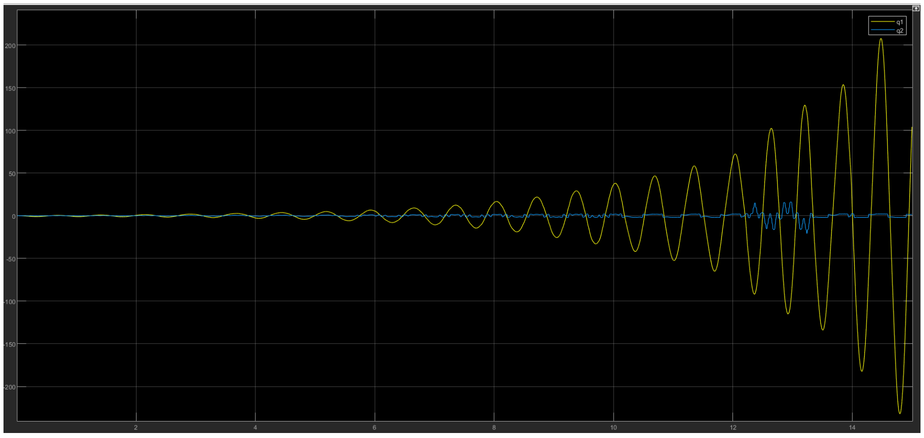

Figure 13: K_{p_1}=60,K_{p_2}=60,K_{i_1}=50,K_{i_2}=50

Keeping K_p constant increasing K_i causes the system to become more unstable as it starts oscillation between more extreme values. But the desired state is achieved more accurately. In a way, if we are able to stablize the system to that desired state,we can achieve the result that we need.

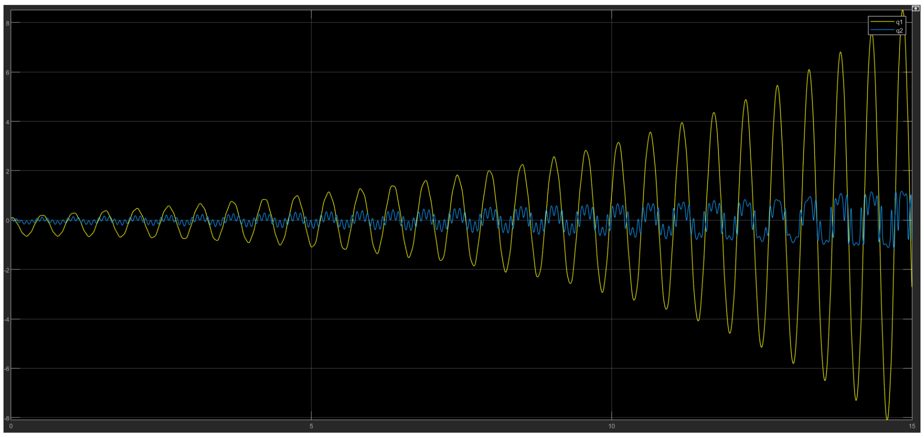

Figure 14: K_{p_1}=120,K_{p_2}=120,K_{i_1}=50,K_{i_2}=50

Increasing K_p, we observe that the rate at which the desired state is achieved increases. In a given period of time, the desired state is achieved more number of times.

7.5 PID Control

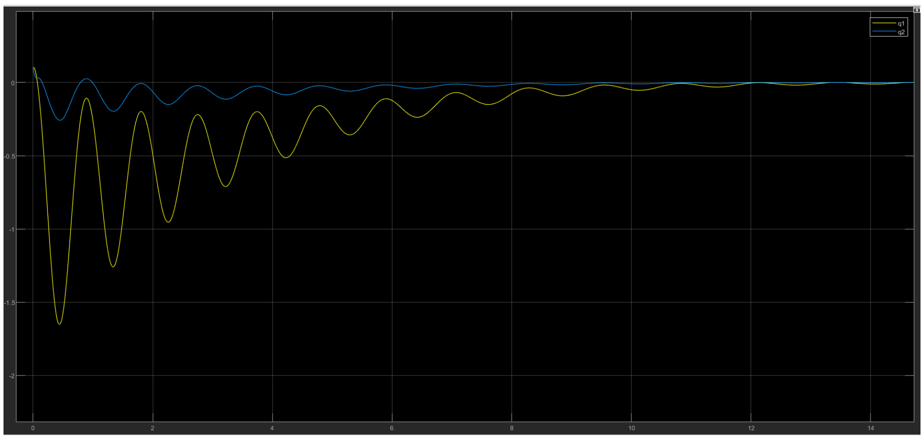

Figure 15: K_{p_1}=20,K_{p_2}=20,K_{d_1}=1,K_{d_2}=1,K_{i_1}=10,K_{i_2}=10

Here, we observe that the steady state is not achieved accurately. The change in output is slow, that is, the transient response is slow. There are too many oscillations though we are slowly advancing towards a steady state.

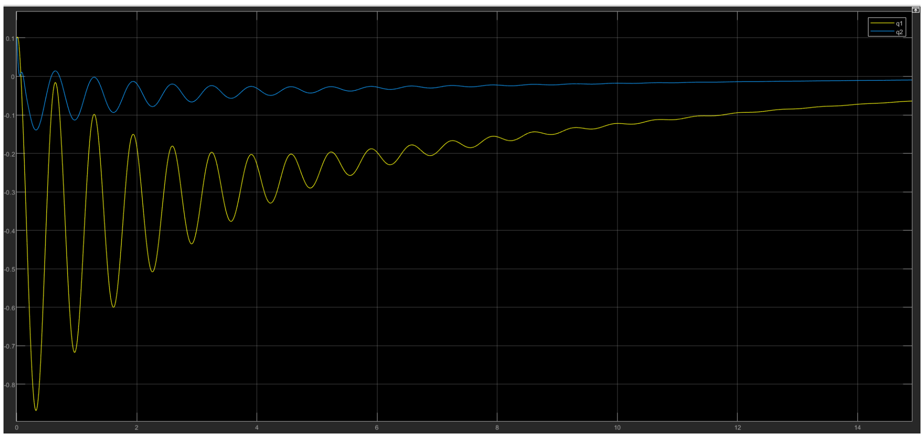

Figure 16: K_{p_1}=70,K_{p_2}=70,K_{d_1}=1,K_{d_2}=1,K_{i_1}=10,K_{i_2}=10

Increasing K_p, we observe that the transient response has improved slightly as in outputs takes a smaller range of angles but the time required for settling increases slightly. The desired state is not achieved correctly . The oscillation also seem a little hectic.

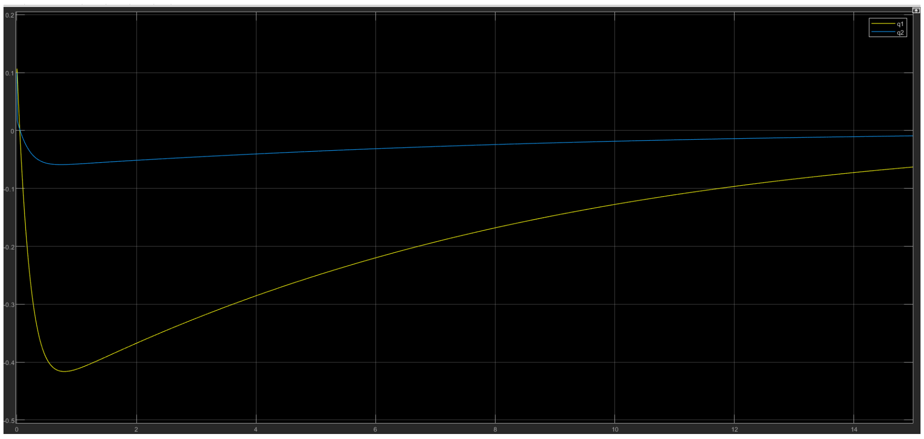

Figure 17: K_{p_1}=70,K_{p_2}=70,K_{d_1}=20,K_{d_2}=20,K_{i_1}=10,K_{i_2}=10

In order to further improve the transient response by decreasing the oscillations, K_d has been increased. Here, we observe a steady state error.

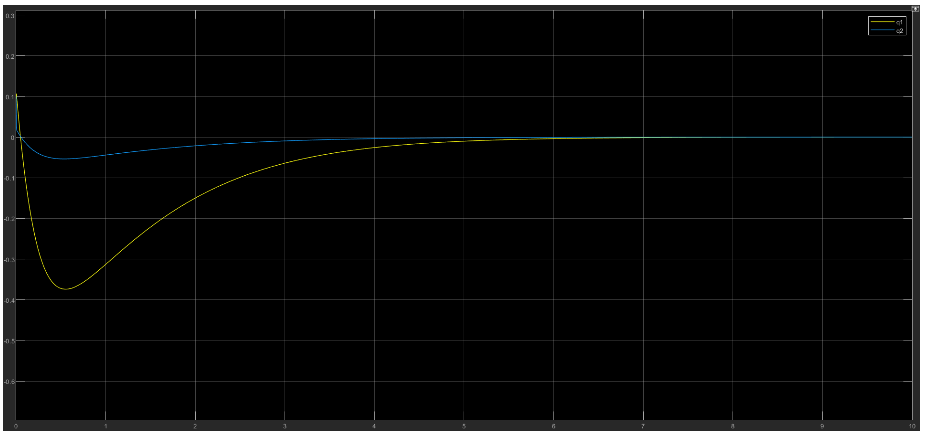

Figure 18: K_{p_1}=70,K_{p_2}=70,K_{d_1}=20,K_{d_2}=20,K_{i_1}=50,K_{i_2}=50

In order to correct the steady state, we are increasing K_i. Here, we observe that the desired steady state is achieved to perfection.

Further tuning the controller to achieve steady state quicker, at the same time sacrificing transient response slightly, we get the following output.

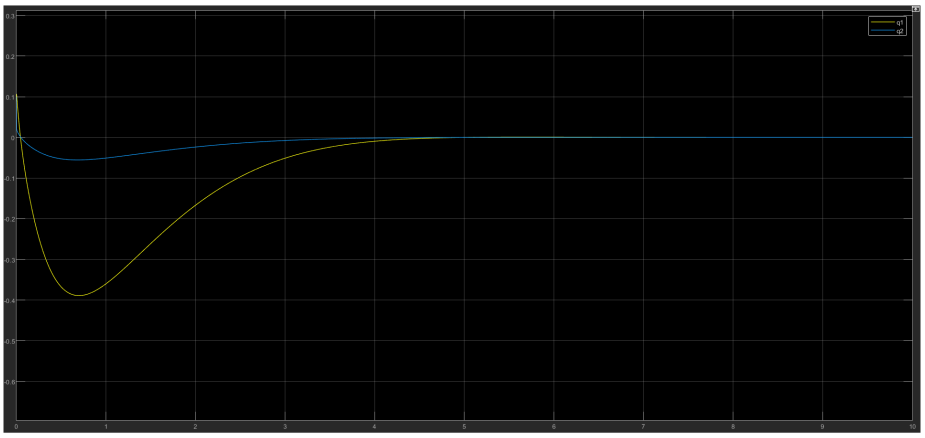

Figure 19: K_{p_1}=60,K_{p_2}=60,K_{d_1}=25,K_{d_2}=25,K_{i_1}=50,K_{i_2}=50

Here, we get the all the characteristics we desired. The output is reaching the desired steady state. The transient response seems good.

For achieving the output we desired,we can use PID controllers with $K_p = 60, K_d = 25, K_i = 50$.

8 Conclusion

Robotic manipulator systems have become integral in manufacturing industries, finding applications in automated architecture machines, multi-capability robots, and drones. Their utilization is driven by the need for speed, accuracy, and repeatability.

Ongoing development focuses on creating new and robust designs that leverage robotic arms to enhance accessibility, precision, and scale. These systems, modeled after human arms, offer insights into complex human movement, particularly through the study of two-link manipulator motion.

This understanding of their movements and design principles empowers production and significantly enhances technical capabilities in various industries.

8.1 Applications

- Automated architecture machines, multi capability robots and drones
- New and robust designs are being developed which deploy the use of robotic arm for advantageous gains in accessibility, precision, scale etc
- Model the working of human arms which helps in understanding the complex movement of human arms after defining the movement of the two-link manipulator

9 References/Bibliography

[1] [2]

References

[1] Mahboub Baccouch and Dodds Stephen. A two-link robot manipulator: Simulation and control design. *International Journal of Robotic Engineering*, 5, 12 2020.

[2] Baccouch Mahboub and Dodds Stephen. A two-link robot manipulator: Simulation and control design. *International Journal of Robotic Engineering*, 5(2), December 2020.