SYSTEMS THINKING

MINI PROJECT

TEAM – DAMAGE CONTROL

TEAM MEMBERS:

- Aayush Das (2022102059)
- Abhishek Sharma (2022102004)
- Akshat Tiwari (2022102043)
- Bhaskar Bhatt (2022102002)
- Dikshant Gurudutt (2022102038)
- Himanshu Gupta (2022102002)
- Himanshu Yadav (2022102010)
- Yash Dusane (2022102078)

Contents:

Question:

Consider the following system dynamics of a 2-link manipulator:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}, \tag{1}$$

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} \\ M_{12} & M_{22} \end{bmatrix}, \mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix},$$

$$M_{11} = (m_1 + m_2)l_1^2 + m_2 l_2 (l_2 + 2l_1 \cos(q_2)),$$

$$M_{12} = m_2 l_2 (l_2 + l_1 \cos(q_2)), M_{22} = m_2 l_2^2,$$

$$\mathbf{C} = \begin{bmatrix} -m_2 l_1 l_2 \sin(q_2)\dot{q}_2 & -m_2 l_1 l_2 \sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0 & m_2 l_1 l_2 \sin(q_2)\dot{q}_2 \end{bmatrix},$$

$$\mathbf{G} = \begin{bmatrix} m_1 l_1 g \cos(q_1) + m_2 g(l_2 \cos(q_1 + q_2) + l_1 \cos(q_1)) \\ m_2 g l_2 \cos(q_1 + q_2) \end{bmatrix}$$

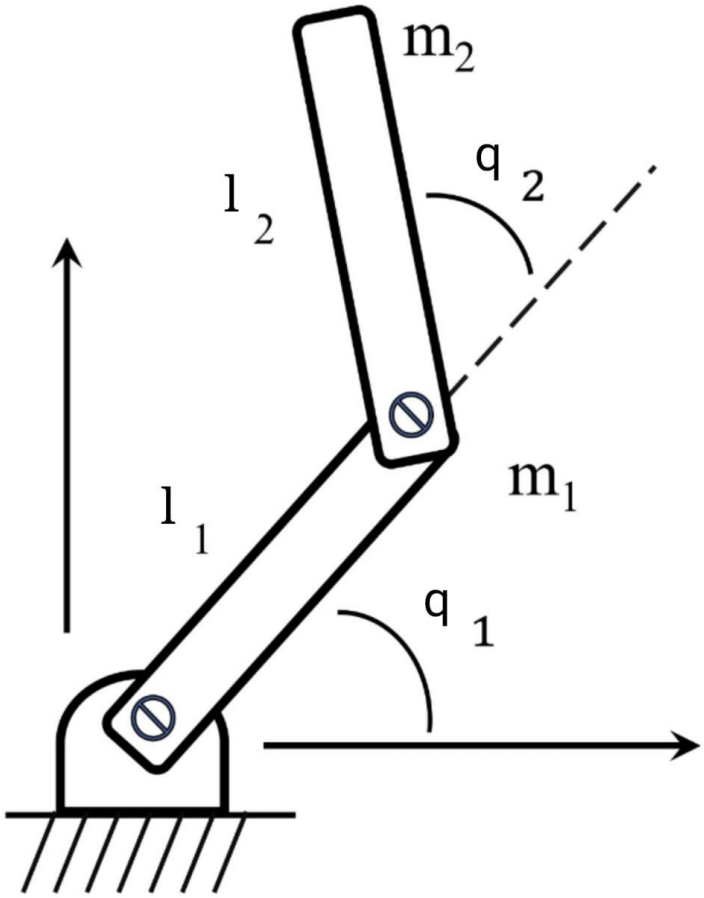
where (m_1, l_1, q_1) and (m_2, l_2, q_2) denote the mass, length and joint angle positions of link 1 and 2 respectively.

The following parametric values are selected: $m_1 = 10$kg, $m_2 = 5$kg, $l_1 = 0.2$m, $l_2 = 0.1$m, $g = 9.81$m/s^2. The joint angles are initially at positions $[q_1(0)\ q_2(0)] = [0.1\ 0.1]$rad.

The objective is to bring the the joint angles from the initial position to $[q_1\ q_2] = [0\ 0]$.

Q. Via MATLAB simulations (choose P, I, and D gains of your choice) show differences in responses (i.e., plot q_1 vs. t and q_2 vs. t) when (i) PD (ii) PI and (iii) PID controllers are applied separately.

Introduction



Above is the system 2-link manipulator which we will be considering in this project. As can be seen it consists of two masses m_1 and m_2 of respective lengths l_1 and l_2. m_1 is at an angle q_1 from the ground and m_2 is at an angle q_2 from the mass m_1. In this project we will be controlling the angles q_1 and q_2 with the help of input torques τ_1 and τ_2 and hence get the desired joint angles with the help of PID controller.

System Implementation:

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

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{12} & M_{22} \end{bmatrix}$$

$$= \begin{bmatrix} (m_1 + m_2)l_1^2 + m_2 l_2(l_2 + 2l_1\cos(q_2)) & m_2 l_2(l_2 + l_1\cos(q_2)) \\ m_2 l_2(l_2 + l_1\cos(q_2)) & m_2 l_2^2 \end{bmatrix}$$

$$C = \begin{bmatrix} -m_2 l_1 l_2 \sin(q_2)\dot{q}_2 & -m_2 l_1 l_2 \sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0 & m_2 l_1 l_2 \sin(q_2)\dot{q}_2 \end{bmatrix}$$

$$G = \begin{bmatrix} m_1 l_1 g cos(q_1) + m_2 g(l_2\cos(q_1 + q_2) + l_1\cos(q_1)) \\ m_2 g l_2\cos(q_1 + q_2) \end{bmatrix}$$

Also m_1 = 10kg, m_2 = 5kg, l_1 = 0.2m, l_2 = 0.1m, g = 9.81m/s^2

$$M = \begin{bmatrix} 0.65 + 0.2\cos(q_2) & 0.05 + 0.1\cos(q_2)) \\ 0.05 + 0.1\cos(q_2)) & 0.05 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.1\sin(q_2)\dot{q}_2 & -0.1\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0 & 0.1\sin(q_2)\dot{q}_2 \end{bmatrix}$$

$$G = \begin{bmatrix} 19.62 cos(q_1) + 49.05(0.1\cos(q_1 + q_2) + 0.2\cos(q_1)) \\ 4.905\cos(q_1 + q_2) \end{bmatrix}$$

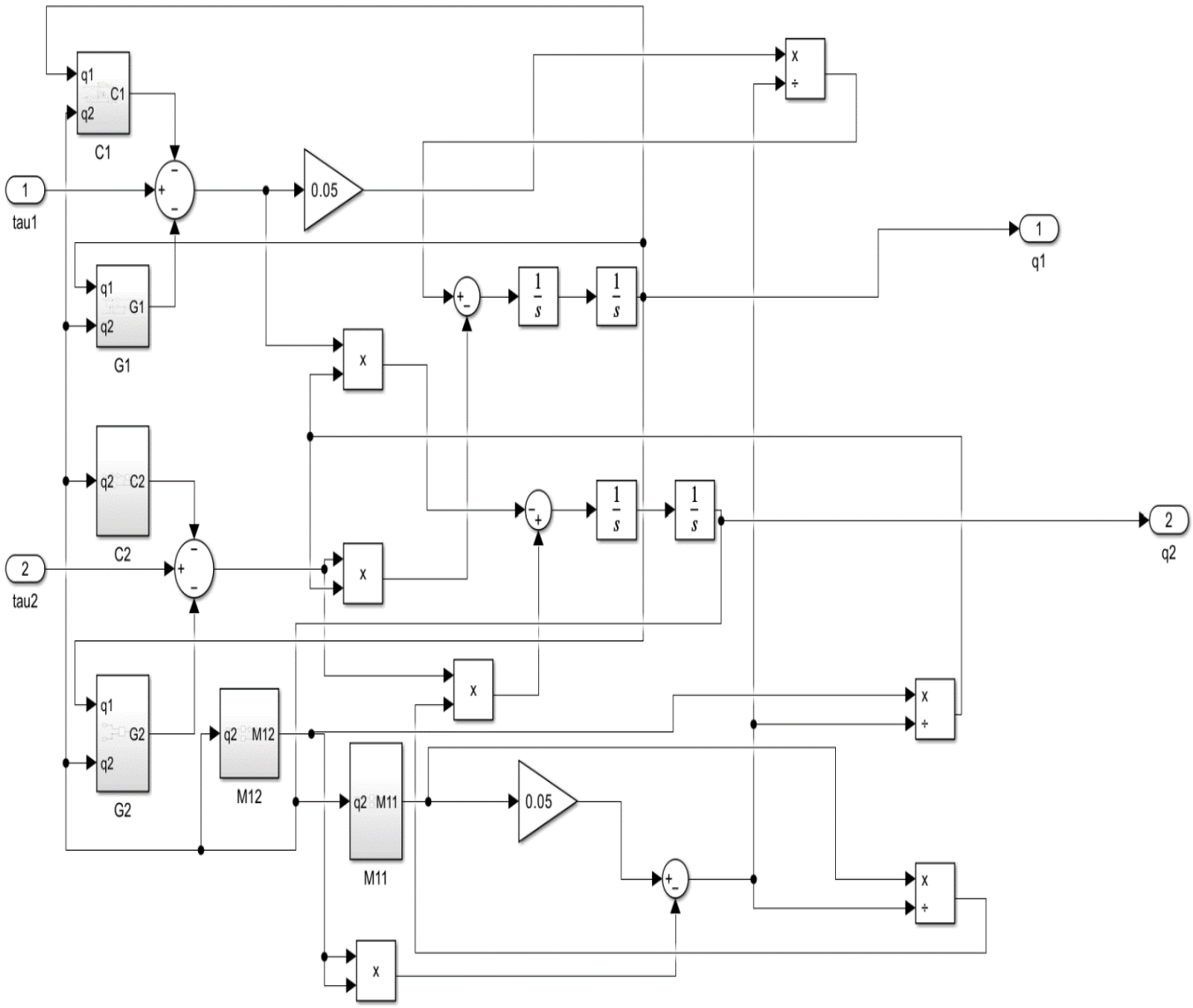
$$\ddot{q} = \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = M^{-1}(\tau - C(q,\dot{q})\dot{q} - G(q))$$

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

$$C' = C(q,\dot{q})\dot{q} = \begin{bmatrix} -0.1\sin(q_2)\dot{q}_2 & -0.1\sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ 0 & 0.1\sin(q_2)\dot{q}_2 \end{bmatrix}\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}$$

$$\ddot{q} = M^{-1}\begin{bmatrix}\tau_1 - C_1' - G_1\\ \tau_2 - C_2' - G_2\end{bmatrix} = \frac{1}{M_{11}M_{22} - M_{12}^2}\begin{bmatrix}M_{22} & -M_{12}\\ -M_{12} & M_{11}\end{bmatrix}\begin{bmatrix}\tau_1 - C_1' - G_1\\ \tau_2 - C_2' - G_2\end{bmatrix}$$

$$q = \begin{bmatrix}q_1\\ q_2\end{bmatrix} = \iint \ddot{q}$$



Implementation by Code

Since

$$\ddot{q} = \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = -M^{-1}\big(C(q,\dot{q})\dot{q} + G(q)\big) + M^{-1}\,\tau$$

Then

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Let's denote the error signals by

$$e(q_1) = q_{1f} - q_1,$$
$$e(q_2) = q_{2f} - q_2$$

Where q_{1f} *and* q_{2f} are the target positions of M_1 *and* M_2 respectively.
Let the initial positions of the M_1 *and* M_2 are $q_1(0)$ *and* $q_2(0)$ respectively.
Then

$$q_0 = \begin{bmatrix} q_1(0) \\ q_2(0) \end{bmatrix}$$

Where q_0 is the initial angle vector.
For using the PID controller let's take the input of the following form –

$$f = K_P e + K_D \dot{e} + K_I \int e\,dt.$$

Where K_P, K_D *and* K_I are the proportional gain, derivative gain, and integral gain respectively.

To control the given system, we can use two independent PID controllers for the two inputs f_1 *and* f_2.
Hence,

$$f_1 = K_{P_1} e_1(q_1) + K_{D_1} \dot{e}_1(q_1) + K_{I_1} \int e_1(q_1)dt$$

$$f_2 = K_{P_2} e_2(q_2) + K_{D_2} \dot{e_2}(q_2) + K_{I_2} \int e_2(q_2) dt$$

$$f_1 = K_{P_1}(q_{1f} - q_1) - K_{D_1} \dot{q_1} + K_{I_1} \int (q_{1f} - q_1) dt \quad (1$$

$$f_2 = K_{P_2}(q_{2f} - q_2) - K_{D_2} \dot{q_2} + K_{I_2} \int (q_{2f} - q_2) dt \quad (2)$$

Since

$$\ddot{q} = \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = -M^{-1}\big(C(q, \dot{q})\dot{q} + G(q)\big) + M^{-1}\tau$$

And in this equation,

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Let the states be,

$$x_1 = \int (q_{1f} - q_1)$$
$$x_2 = \int (q_{2f} - q_2)$$
$$x_3 = q_1$$
$$x_4 = q_2$$
$$x_5 = \dot{q_1}$$
$$x_6 = \dot{q_2}$$

Then the derivatives of the states will be,

$$\dot{x_1} = q_{1f} - q_1 = q_{1f} - x_3$$
$$\dot{x_2} = q_{2f} - q_2 = q_{2f} - x_4$$
$$\dot{x_3} = \dot{q_1} = x_5$$
$$\dot{x_4} = \dot{q_2} = x_6$$
$$\dot{x_5} = \ddot{q_1}$$
$$\dot{x_6} = \ddot{q_2}$$

Where $\ddot{q_1}$ *and* $\ddot{q_2}$ can be written in terms of state variables using the following equation.

$$\ddot{q} = \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = -M^{-1}\big(C(q, \dot{q})\dot{q} + G(q)\big) + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \quad (3)$$

Now we have a first order differential equation in terms of states as follows.

$$\dot{X} = \frac{dX}{dt} = \begin{bmatrix} q_{1f} - x_3 \\ q_{2f} - x_4 \\ x_5 \\ x_6 \\ \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} \tag{4}$$

Where $\dot{X}$ is the state derivatives vector for an unknown state vector X.
Let the initial state vector be X_0

$$X_0 = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \\ x_5(0) \\ x_6(0) \end{bmatrix}$$

∵ Given initial values of q_1 *and* q_2 as 0.1 rad and 0.1 rad respectively.
∴ $x_3(0)$ *and* $x_4(0)$ are equal to 0.1 rad and 0.1 rad respectively.
∵ Initially the integral interval is almost equal to zero hence the values of $x_1(0)$ *and* $x_2(0)$ are zero.
∵ The derivative of $x_3(0)$ *and* $x_4(0)$ is equal to zero hence the values of $x_5(0)$ *and* $x_6(0)$ are also zero.
Hence the initial state vector for the given system is as follows.

$$X_0 = \begin{bmatrix} 0 \\ 0 \\ 0.1 \\ 0.1 \\ 0 \\ 0 \end{bmatrix}$$

Now we have state derivatives vector ($\dot{X}$) having values in terms of state variables and the initial state vector (X_0).
So, we can solve for the unknown vector X using the MATLAB function ode45.

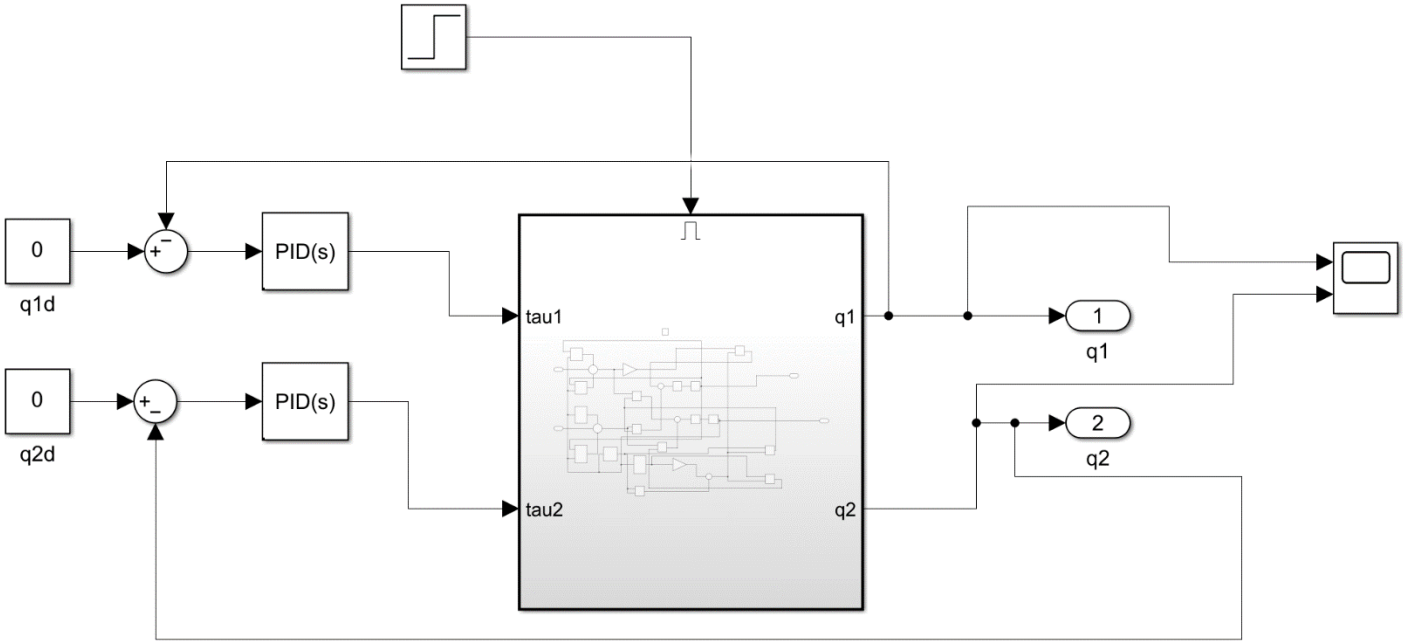
Code Explanation

Project.m

```
% Time Grid
t = [0 30];

% Initial State Vector
X0 = [0 0 0.1 0.1 0 0];

[t,Y] = ode45(@statederivatives,t,X0);

% Plot of q1 and q2 vs Time
figure(1)
plot(t,Y(:,3),'r',t,Y(:,4),'b')
title('q_1 and q_2 vs Time')
xlabel('Time (s)')
ylabel('q_1 and q_2 (rad)')
legend('q_1','q_2')
```

Here in t is a vector having values between 0 to 30 with very fine steps. ode45 is taking a function (state derivatives) and time grid (t) and initial state vector ($X0$) and returning a vector having values same as time grid(t) and a matrix of dimension (length of t × 6) named Y. Here Y have n (equal to length of t) number of rows where i^{th} row corresponds to the value of state vector for i^{th} value of time grid(t). In plot function we are plotting the 3^{rd} and 4^{th} value of the state vector for all rows in the matrix Y vs time grid (t) which is the value of the q_1 *and* q_2.
State derivatives function calculates the state derivatives vector ($\dot{X}$) for any state vector X using the equation (1), (2), (3) and (4).

PID Implementation in Simulink:



Above is a PID controlled two-link manipulator system implemented in Simulink where the enabled subsystem is the plant for this model with the given inputs and outputs. And the inputs to PID controllers are the error between the desired output and present output at every state. The plant is an enabled subsystem because we must keep initial output variables to a non-zero value and for that purpose an enabled subsystem is used for plant in this model.

Prerequisite Definitions:

- **Maximum Overshoot:** The difference between the peak of first time and steady output is called maximum overshoot.

- **Settling Time:** The time that is required for the response to reach and stay within the specified range (2% to 5%) of its final value is called the settling time.

- **Peak Time:** The time required for the response to reach the 1st peak of the time response or 1st peak overshoot is called the Peak time.

- **Steady State Error:** The difference between actual output and desired output as time tends to infinity is called the steady state error of the system.

PID

A feedback control system that maintains or regulates a desired setpoint (target value) by continuously adjusting a control output based on the error between the desired setpoint and the measured process variable.

$$f = K_P e + K_D \dot{e} + K_I \int e\, dt.$$

Where K_P, K_D *and* K_I are the proportional gain, derivative gain, and integral gain respectively.

Theory:
proportionality Gain (K_P):

- Increasing K_P decreases Rise Time.
- Increasing K_P for a faster response may introduce instability or excessive oscillations.

Proportionality Integral (K_I):

- Increasing K_I Eliminates Steady State Error.
- Integral terms can accumulate large errors over time and cause large and delayed responses.

Proportionality Derivative (K_D):

- Uses damping oscillations to decrease maximum Overshoot.

Steady State Error = 0 because integral component sums the error term over time. The result is that small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero.

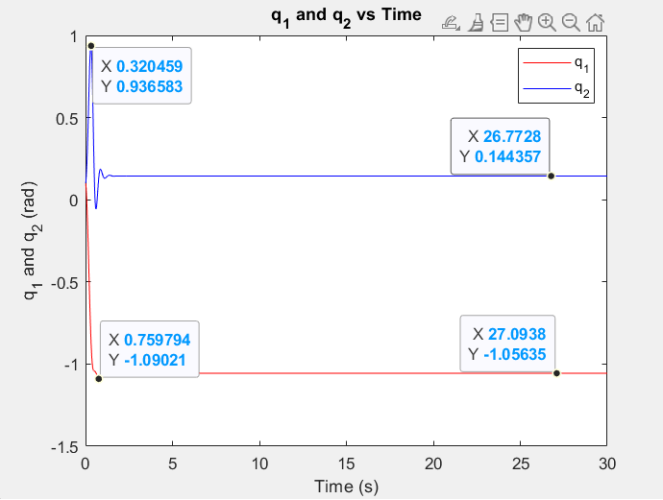
Advantages:

- Simple and easy to implement.
- Fast and Stable response as they can eliminate steady state error, minimize Overshoot and oscillations.
- Can handle a wide range of system dynamics and disturbance.

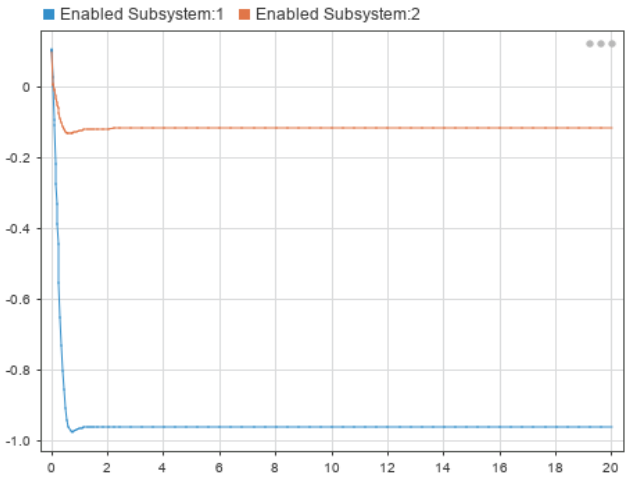
Observations – PD:

$K_{P_1} = 20, K_{I_1} = 0, K_{D_1} = 10$
$K_{P_2} = 20, K_{I_2} = 0, K_{D_2} = 10$
Here we have
1.Peak Value = 0.9 rad, Peak Time = 0.32 sec, Settling Time = 2 sec
2.Peak Value = -1.1 rad, Peak Time = 0.75 sec, Settling Time = 1.20sec
Here we can observe that it first reaches the peak value in the peak time mentioned above then settles to a steady state with error. So, we have some steady state errors. The steady state error is 0.15 in the case of plot 1 and -1.05 in case of plot 2.

MATLAB Plot

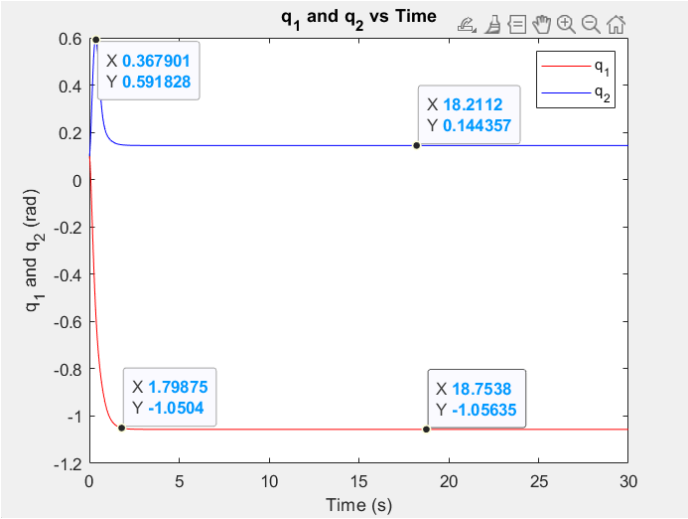


Simulink Plot

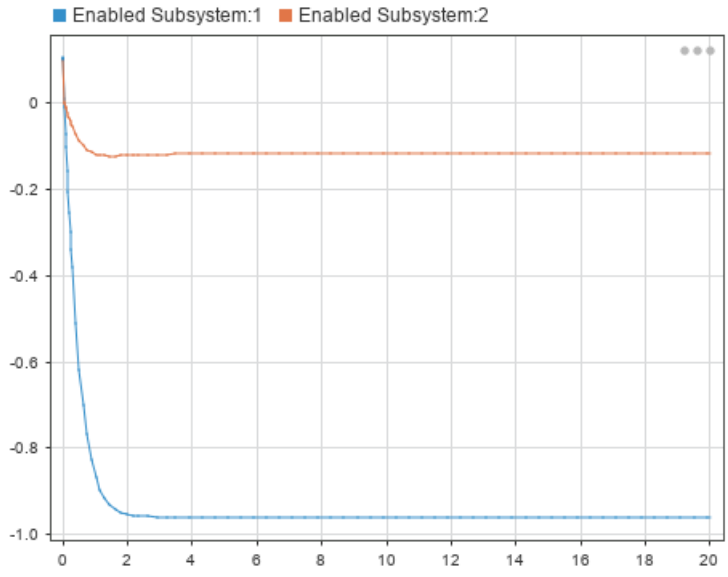


$K_{P_1} = 20, K_{I_1} = 0, K_{D_1} = 20$
$K_{P_2} = 20, K_{I_2} = 0, K_{D_2} = 20$
Here we have
1.Peak Value = 0.6rad, Peak Time = 0.36 sec, Settling Time = 2.00sec
2.Peak Value = -1.05rad, Settling Time = 1.20sec
Here we can observe that it first reaches the peak value in the peak time mentioned above then settles to a steady state with error. So, we have some steady state errors. The steady state error is 0.15 in the case of plot 1 and -1.05 in case of plot 2. Since the value of K_D is increased hence the peak value (overshoot) decreased as compared to previous case.

MATLAB Plot

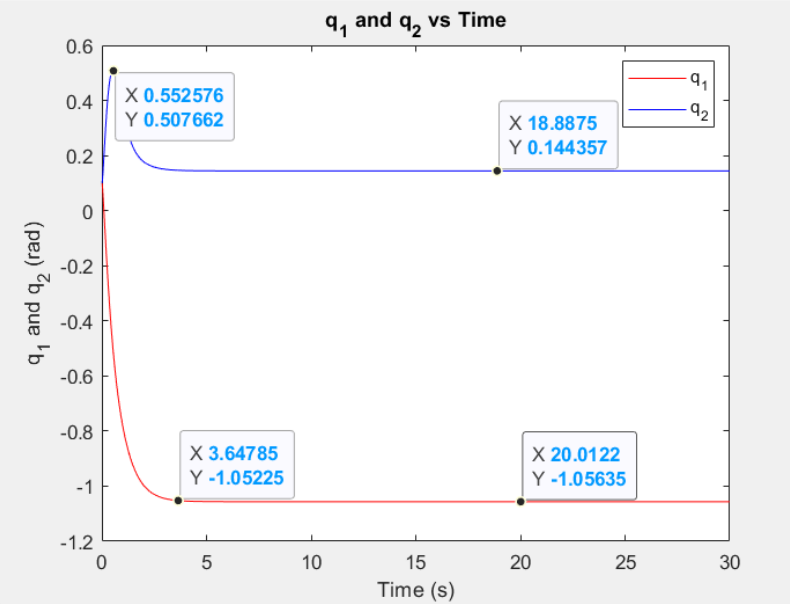


Simulink Plot

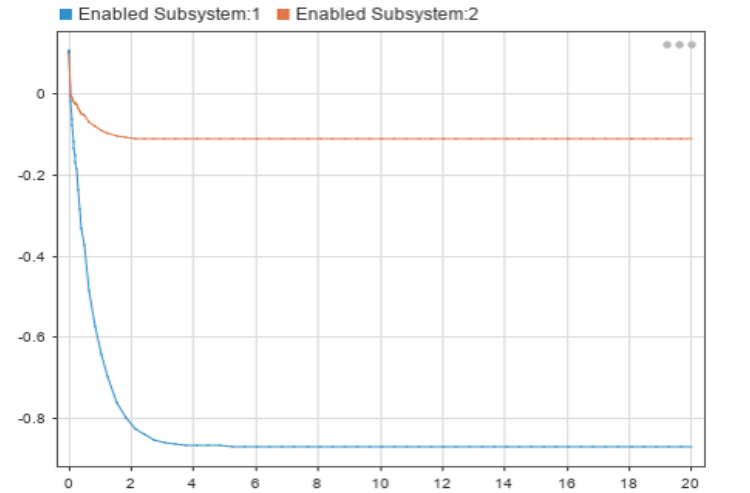


$K_{P_1} = 20, K_{I_1} = 0, K_{D_1} = 35$
$K_{P_2} = 20, K_{I_2} = 0, K_{D_2} = 35$
Here we have
1.Peak Value = 0.5 rad, Peak Time = 0.55 sec, Settling Time = 2.00sec
2.Peak Value = -1.05 rad, Peak Time = 0.9 sec, Settling Time = 3.50sec
Here we can observe that it first reaches the peak value in the peak time mentioned above then settles to a steady state with error. So, we have some steady state errors. The steady state error is 0.16 in the case of plot 1 and -1.05 in case of plot 2. Since the value of K_D is increased hence the peak value (overshoot) decreased as compared to previous case. The peak value is also achieved much faster.

MATLAB Plot

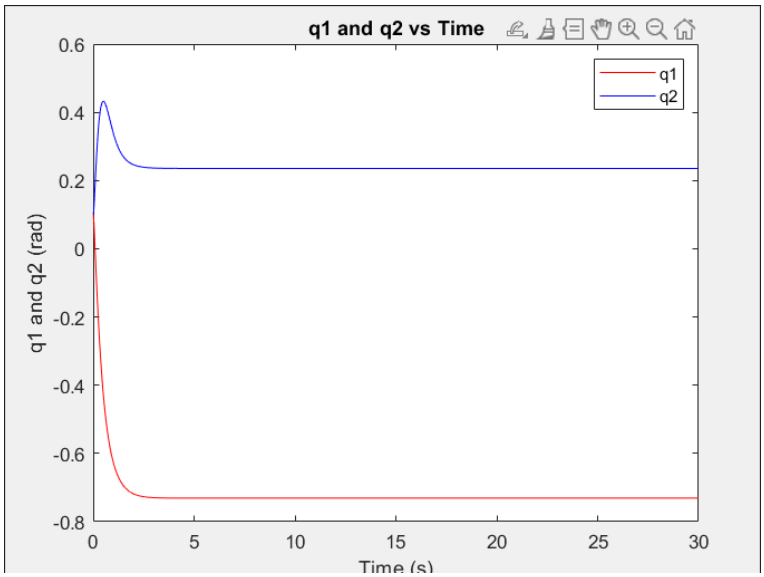


Simulink Plot

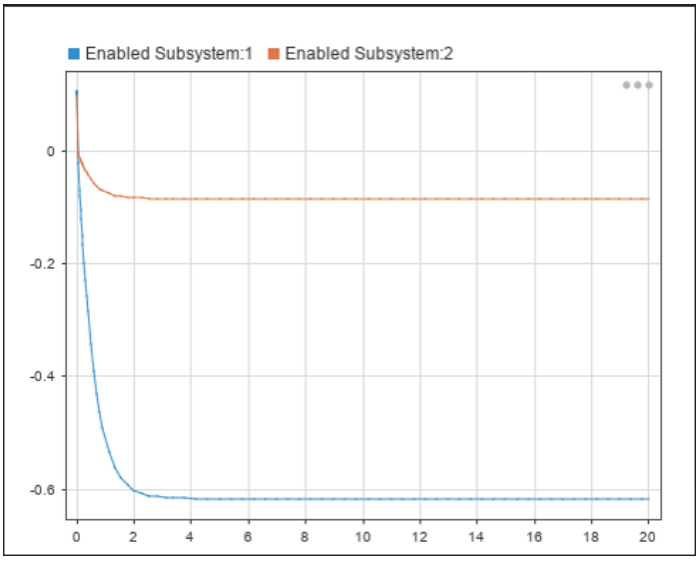


$K_{P_1} = 45, K_{I_1} = 0, K_{D_1} = 35$
$K_{P_2} = 45, K_{I_2} = 0, K_{D_2} = 35$
Here we have
1.Peak Value = 0.42rad, Peak Time = 1.2 sec, Settling Time =2.5sec
2.Peak Value = -0.65rad, Peak Time = 2.5 sec, Settling Time = 2.5sec
Here we can observe that it first reaches the peak value in the peak time mentioned above then settles to a steady state with error. So, we have some steady state errors. The steady state error is 0.41 in case of plot 1 and -0.7 in case of plot 2. The peak value decreases in both cases. Increasing K_P reduces the steady state error as we can observe from the plot.

MATLAB Plot

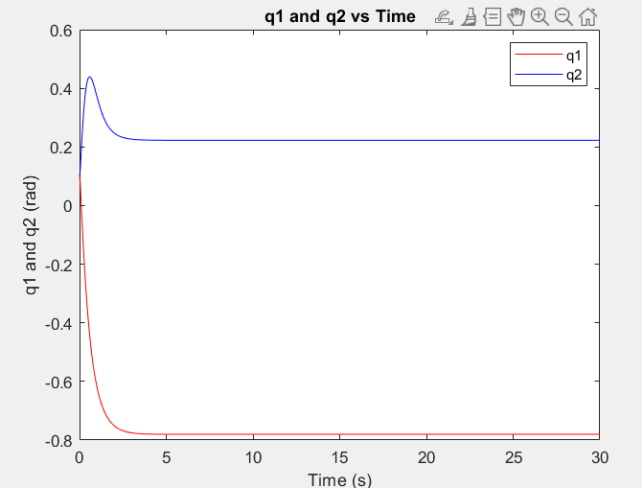


Simulink Plot

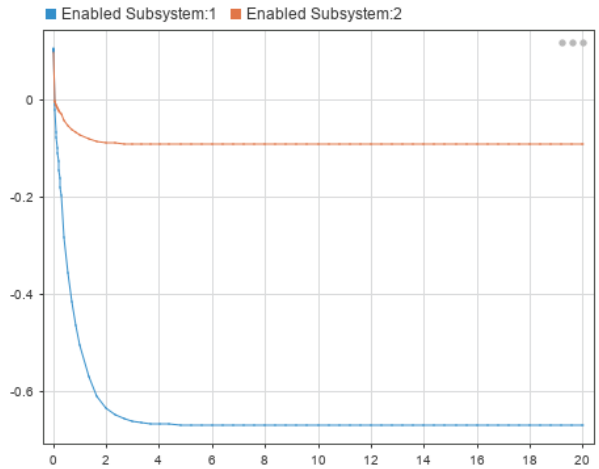


$K_{P_1} = 40, K_{I_1} = 0, K_{D_1} = 40$
$K_{P_2} = 40, K_{I_2} = 0, K_{D_2} = 40$
Here we have
1.Peak Value = 0.41rad, Peak Time = 0.8 sec, Settling Time =4.00sec
2.Peak Value = -0.9rad, Peak Time = 3.0 sec, Settling Time = 3.0sec
Here we can observe that it first reaches the peak value in the peak time mentioned above then settles to a steady state with error. So, we have a finite steady state error. The steady state error is 0.4 in case of plot 1 and -0.75 in case of plot 2. The peak value decreases in both cases. Increasing K_P reduces the steady state error as we can observe from the plot.

MATLAB Plot



Simulink Plot

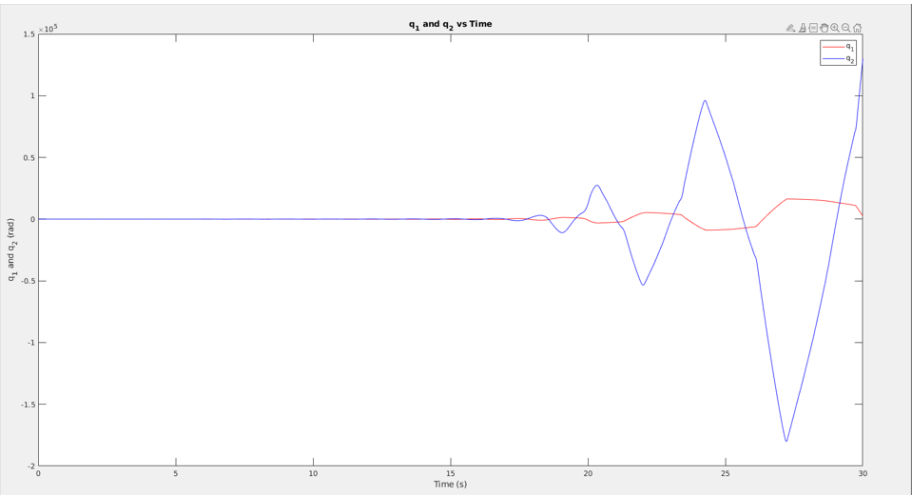


Observations – PI:

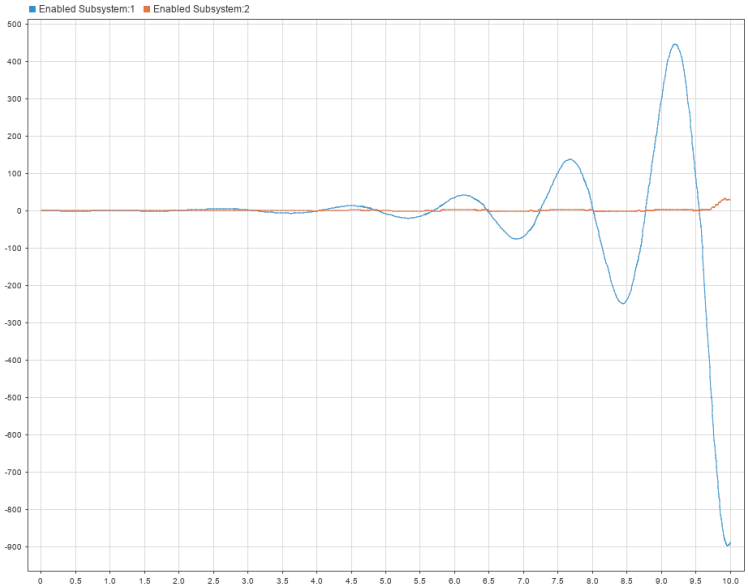
$K_{P_1} = 10, K_{I_1} = 10, K_{D_1} = 0$
$K_{P_2} = 10, K_{I_2} = 10, K_{D_2} = 0$

Here we can observe that due to the lack of derivative control ($K_D = 0$), the system may become prone to oscillations over time. This means that after the initial stable response, there will be oscillations around the setpoint. These oscillations can occur because the controller doesn't consider the rate of change of the error, which can lead to overshooting and corrective actions that keep switching back and forth.

MATLAB Plot



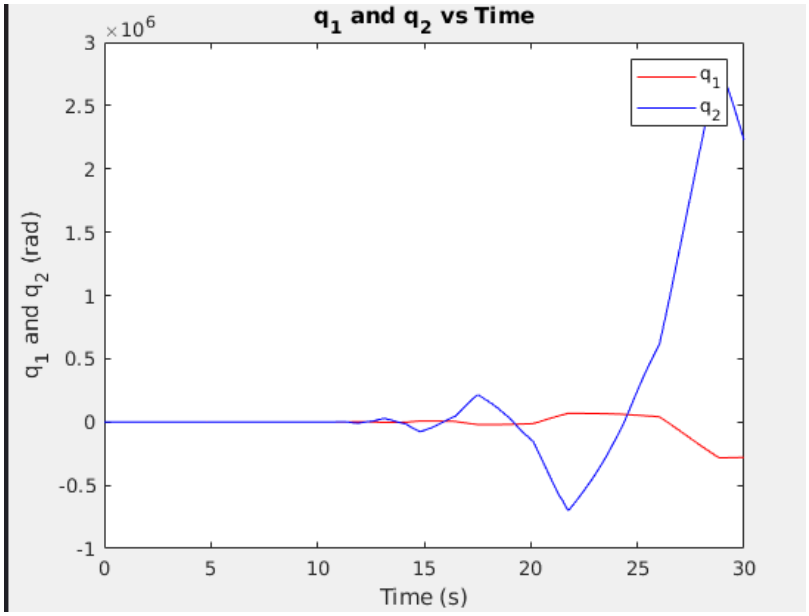
Simulink Plot



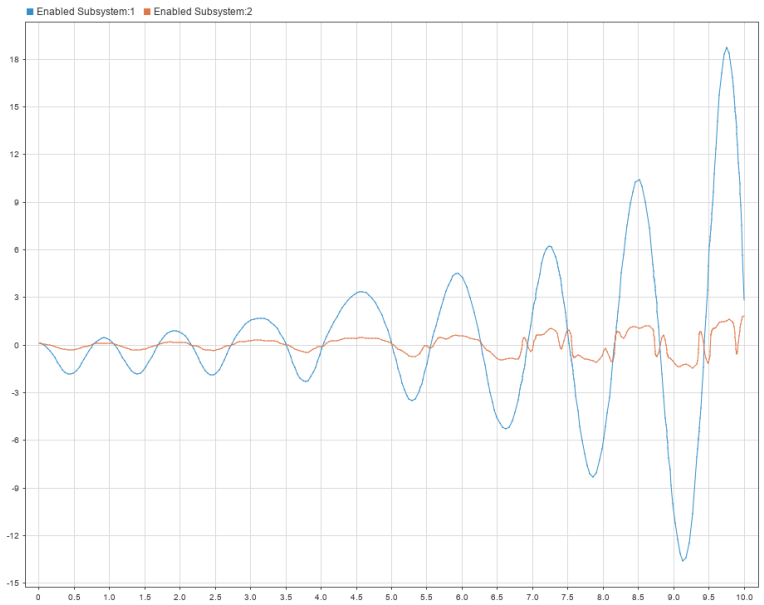
$K_{P_1} = 20, K_{I_1} = 10, K_{D_1} = 0$
$K_{P_2} = 20, K_{I_2} = 10, K_{D_2} = 0$

In this case, K_P is increased from previous value and hence system response is faster than previous case. Since K_D is still zero, high magnitude of overshooting and undershooting can also be observed.

MATLAB Plot



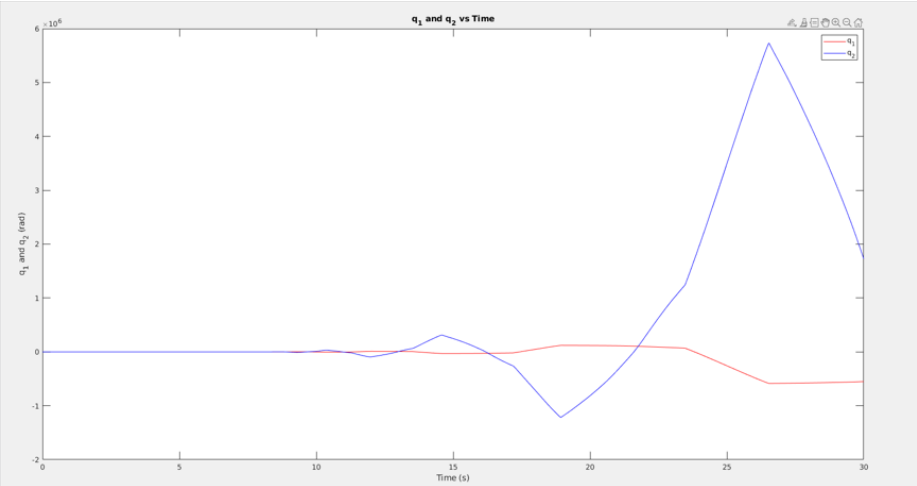
Simulink Plot



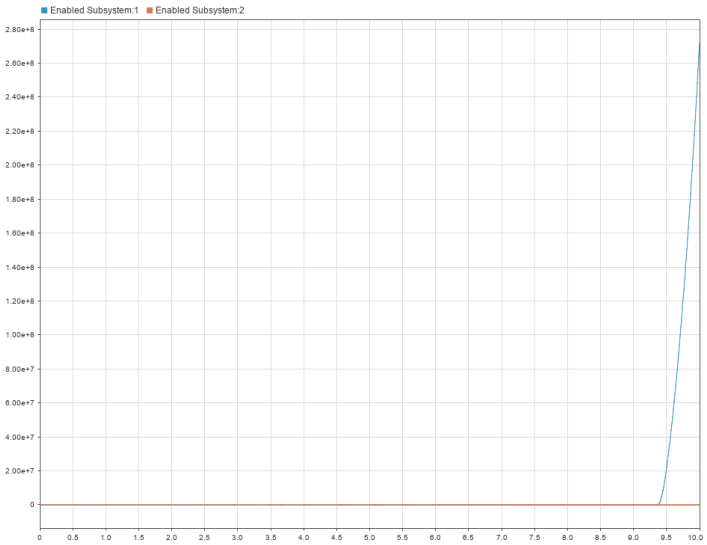
$K_{P_1} = 25, K_{I_1} = 35, K_{D_1} = 0$
$K_{P_2} = 25 K_{I_2} = 35, K_{D_2} = 0$

Here we can observe that the system eliminates the steady state error more effectively because the value of K_P is increased but the response is slower. The system becomes more aggressive, and we can observe oscillating unstable output due to K_D being zero.

MATLAB Plot



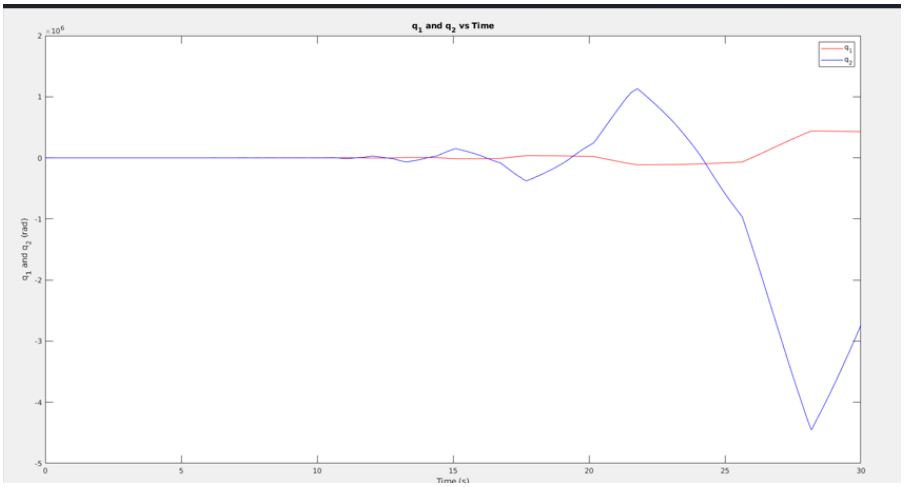
Simulink Plot



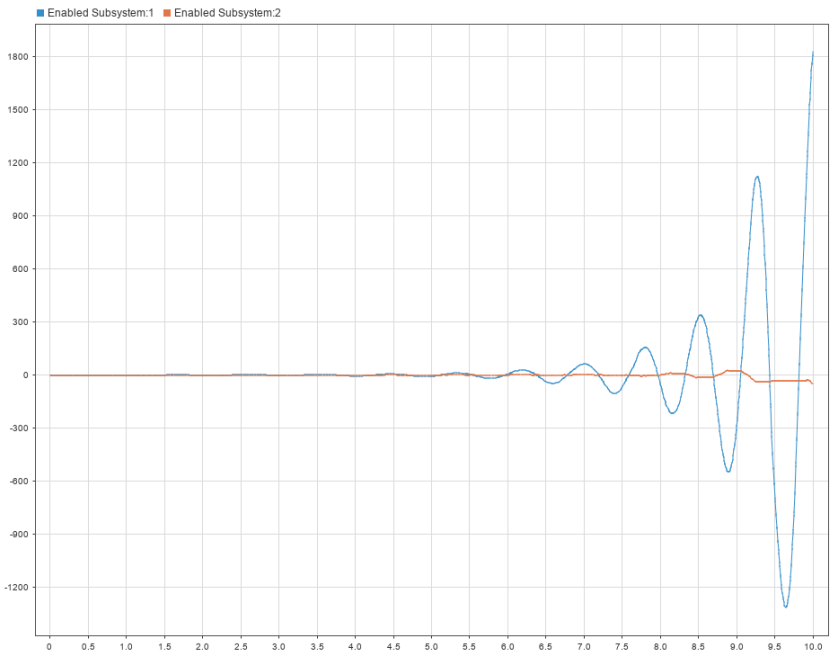
$K_{P_1} = 40, K_{I_1} = 40, K_{D_1} = 0$
$K_{P_2} = 40\ , K_{I_2} = 40, K_{D_2} = 0$

Here we can observe that high value of both K_P and K_I results in fast initial response of the system and the overshooting also increases due to strong integral action. The systems oscillate and are not stable.

MATLAB Plot



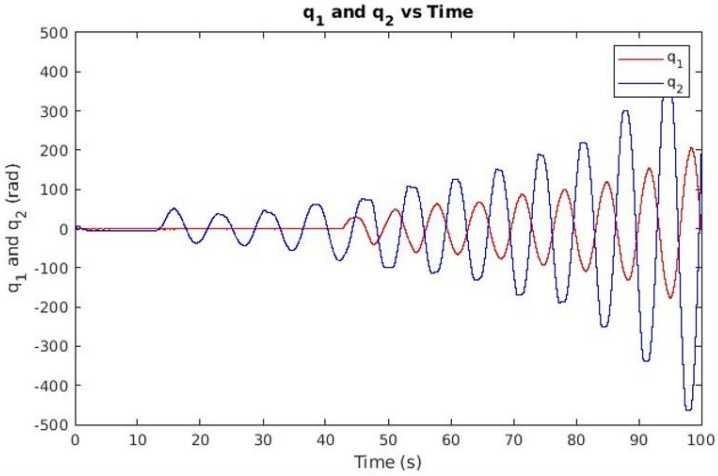
Simulink Plot



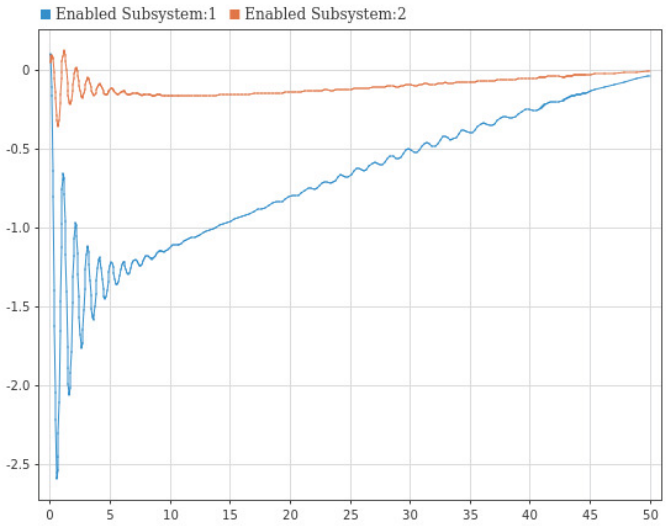
Observations – PID:

$K_{P_1} = 1, K_{I_1} = 1, K_{D_1} = 1$
$K_{P_2} = 1, K_{I_2} = 1, K_{D_2} = 1$
Here we have
1.Peak Value = -2.6 rad, Peak Time = 0.6 sec
2.Peak Value = -0.35 rad, Peak Time = 0.6 sec
Oscillation and undershoot can be seen clearly in transient response, The K_I value leads for a finite settling time, unlike a PD controller system. But K_P and K_D don't affect the error much. We can see oscillations dominant till 10sec.

MATLAB Plot

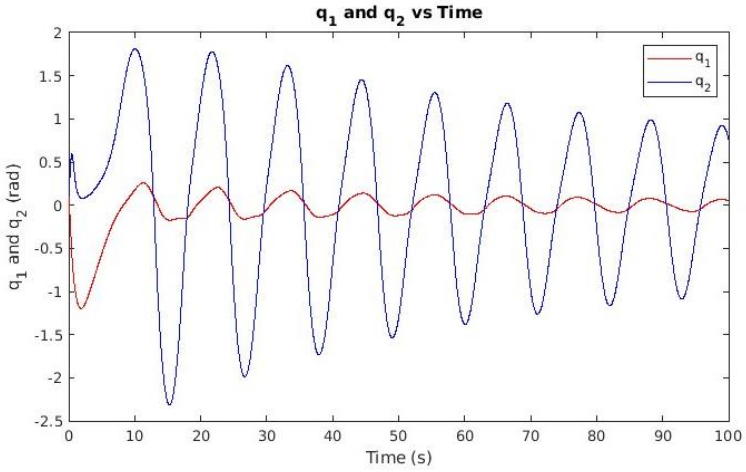


Simulink Plot

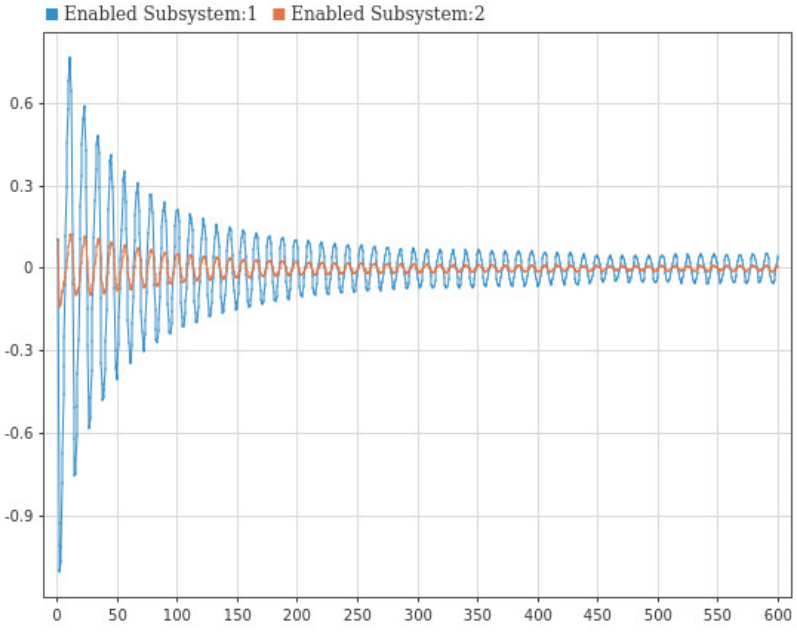


$K_{P_1} = 1, K_{I_1} = 10, K_{D_1} = 30$
$K_{P_2} = 1, K_{I_2} = 10, K_{D_2} = 30$
Here we have
1.Peak Value = -1.1 rad, Peak Time = 2 s
2.Peak Value = -0.15 rad, Peak Time = 2 s
In this case, we can see oscillations dominant all the time, even in steady state. This is due to the lower value of K_P.
The overshoot and undershoot has decreased with increase in K_D.
K_I is giving steady state error 0, but smaller value of K_P is causing oscillating.

MATLAB Plot

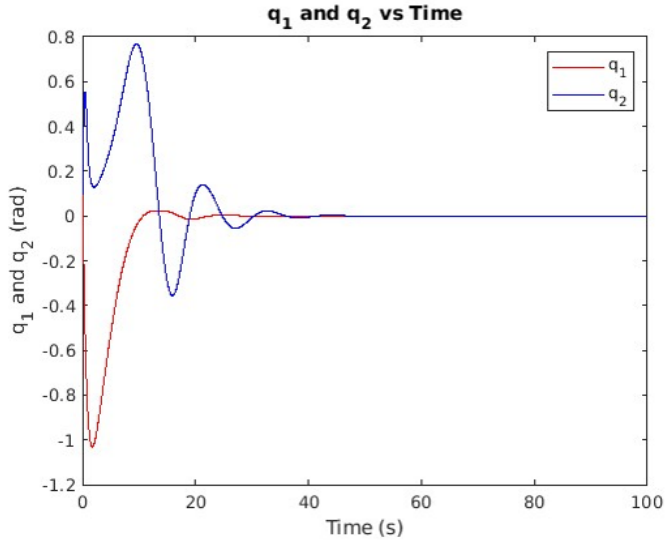


Simulink Plot

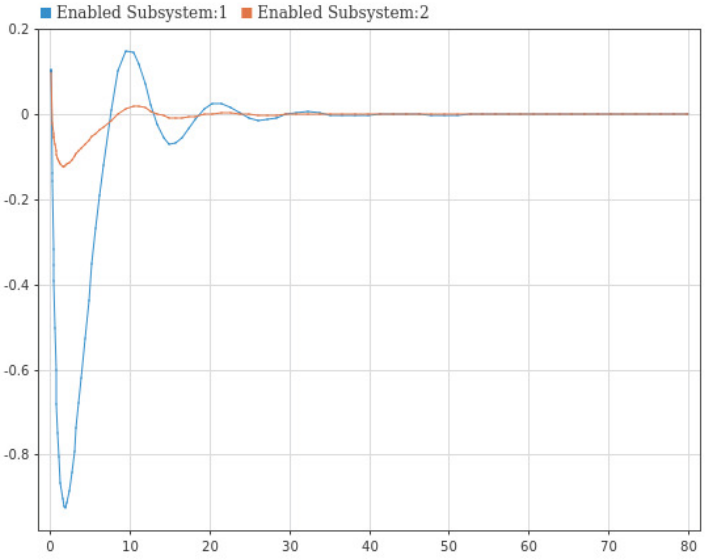


$K_{P_1} = 10, K_{I_1} = 10, K_{D_1} = 30$
$K_{P_2} = 10, K_{I_2} = 10, K_{D_2} = 30$
Here we have
1.Peak Value = -0.9 rad, Peak Time = 1.7 s, Settling Time = 40 s
2.Peak Value = -0.1 rad, Peak Time = 1.7 s, Settling Time = 40 s
Now as we increase the value of K_P, then the oscillations have reduced.
Thus, K_I could also make steady state error 0.
Also, we can see settling time reducing to 40 s.
We can see undershoot initially, but then the overshoot and undershoot gradually decreased.

MATLAB Plot

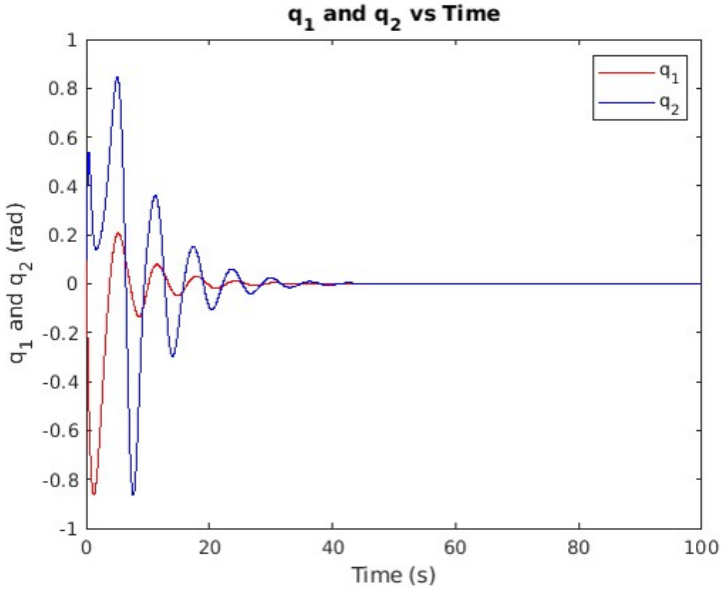


Simulink Plot

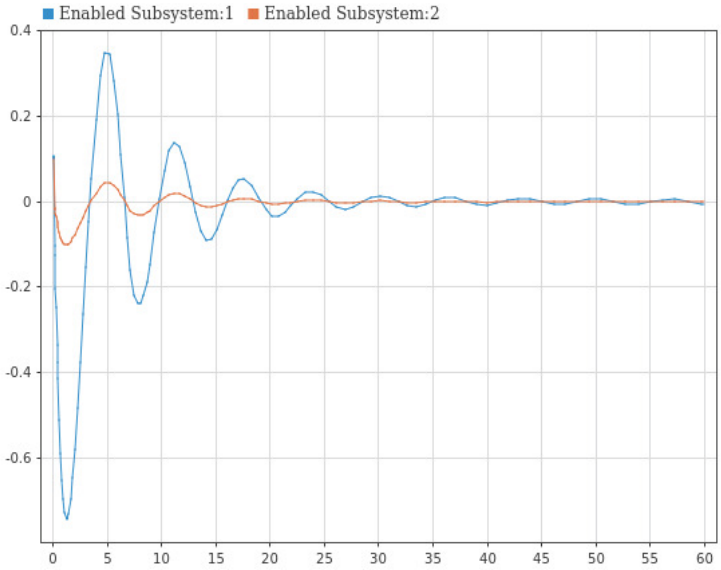


$K_{P_1} = 10, K_{I_1} = 30, K_{D_1} = 30$
$K_{P_2} = 10, K_{I_2} = 30, K_{D_2} = 30$
Here we have
1.Peak Value = -0.7 rad, Peak Time = 1.2 s, Settling Time = 40 s
2.Peak Value = -0.1 rad, Peak Time = 1.2 s, Settling Time = 40 s
Here, we changed the K_I, but this didn't affect the plot much. As K_I only makes the steady state error 0. There isn't a difference in settling time too.
But the oscillations have increased, though decrease in Peak Value.

MATLAB Plot



Simulink Plot

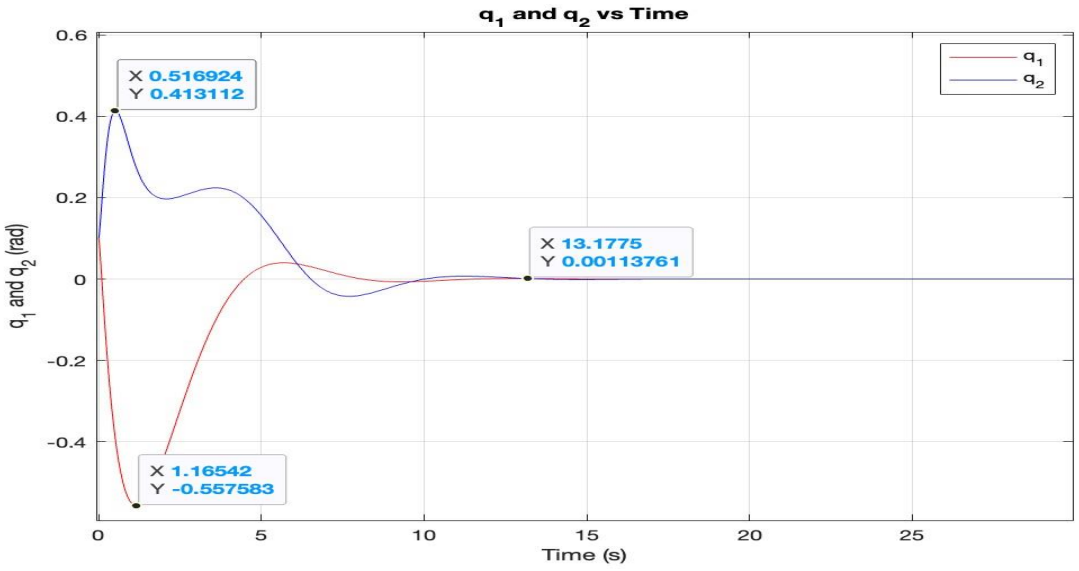


$K_{P_1} = 40,\ K_{I_1} = 40, K_{D_1} = 40$
$K_{P2} = 40, K_{I2} = 40, K_{D1} = 40$

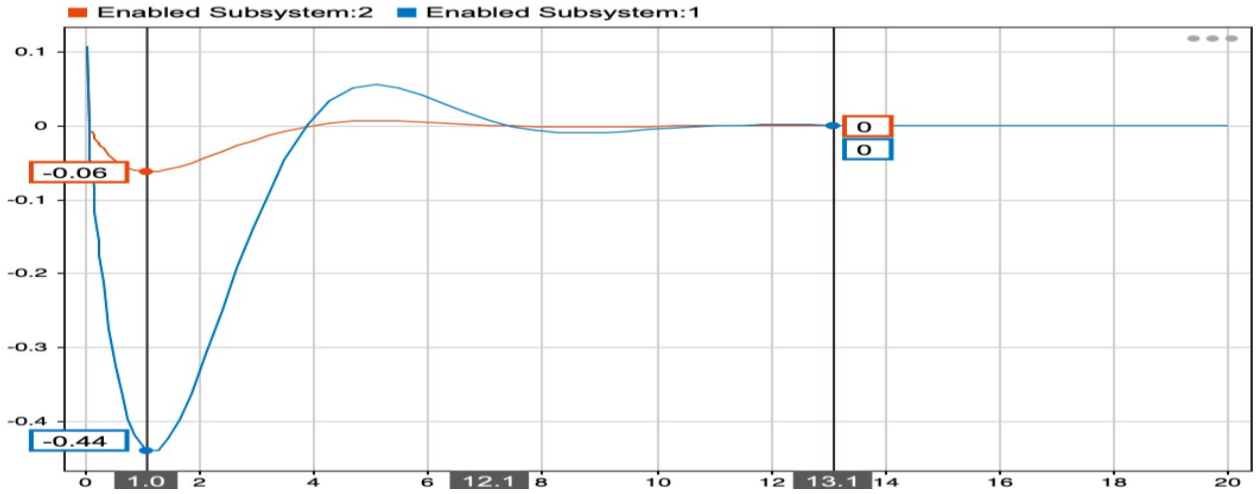
Here we have
1.Peak Value = -0.557 rad, Peak time = 1.16 s, Settling Time = 13.1775 s
2.Peak Value = 0.413 rad, Peak time = 0.413 s, Settling Time = 13.1775 s

The K_I value leads for a finite settling time, like a PD controller system. Settling time is reached instantaneously because of large K_P values.

MATLAB Plot



Simulink Plot

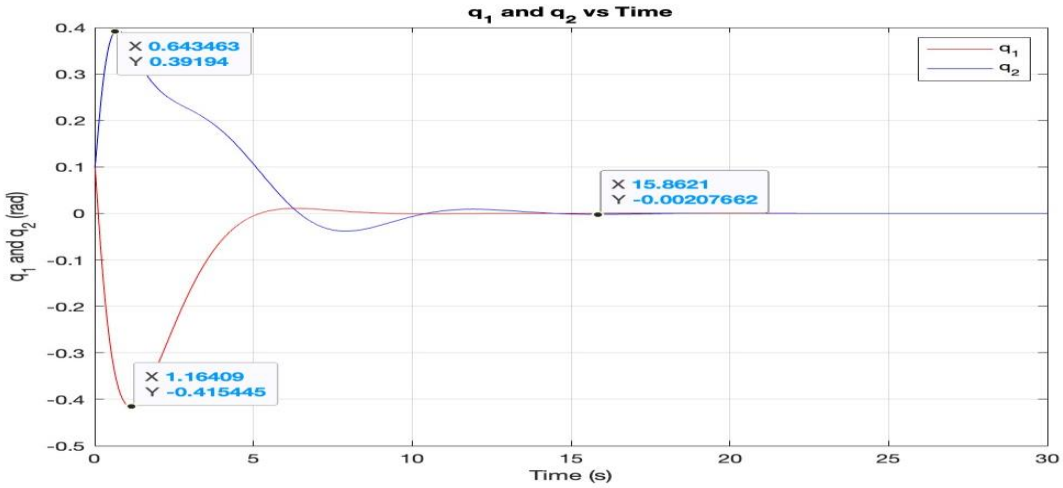


$K_{P_1} = 50, K_{I_1} = 75, K_{D_1} = 66$
$K_{P_2} = 40, K_{I_2} = 40, K_{D_2} = 55$

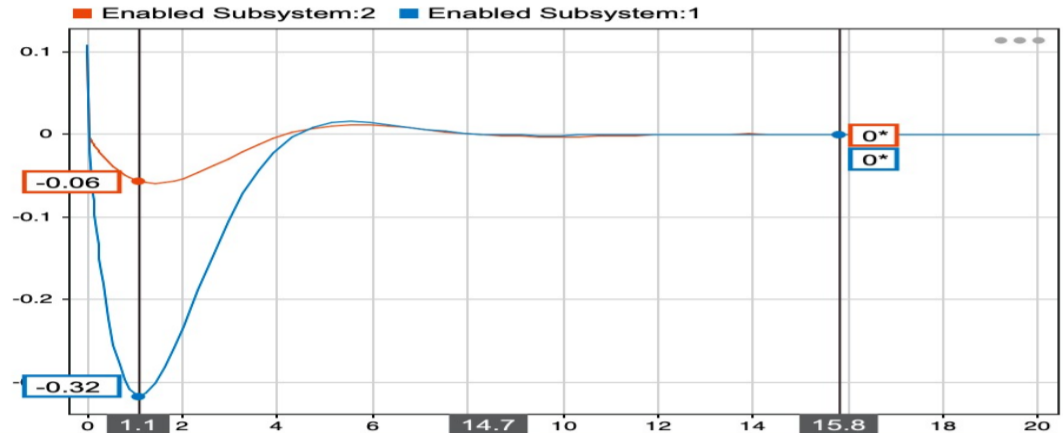
Here we have
1.Peak Value = -0.415 rad, Peak time = 1.164 s, Settling Time = 15.8621 s
2.Peak Value = 0.391 rad, Peak time = 0.643 s, Settling Time = 15.8621 s

In system 2, we kept K_P and K_I fixed, and we increased K_D, because of this Maximum overshoot reduces and Settling time increases.
In system 1, we increase K_P, K_I and K_D, because of increase in K_D the settling time increases and maximum overshoot decreases. By increasing K_P, the number of oscillations decreases. Because of non-zero value of K_I we get zero steady state error in PID system.

MATLAB Plot



Simulink Plot

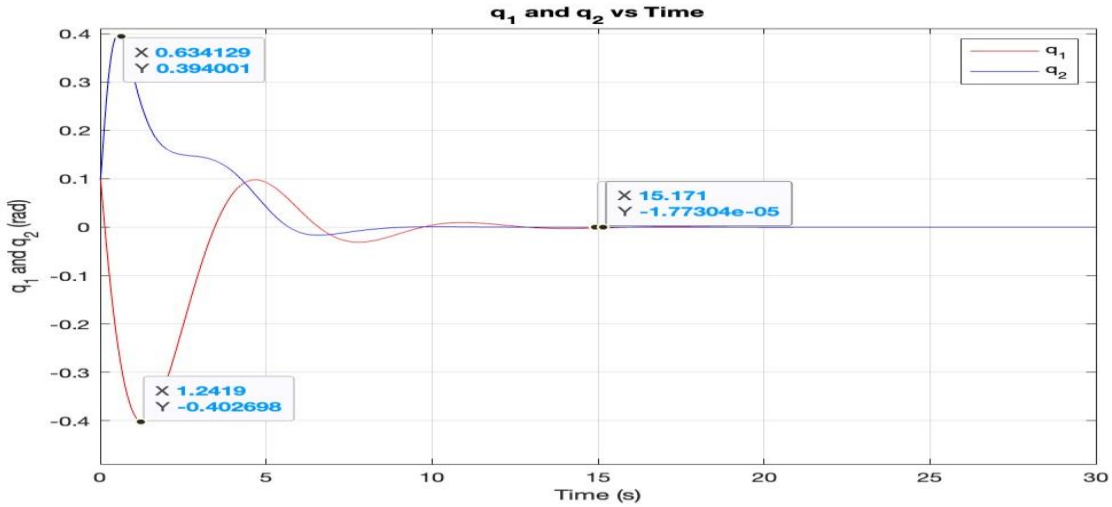


$K_{P_1} = 66, K_{I_1} = 50, K_{D_1} = 50$
$K_{P_2} = 62, K_{I_2} = 55, K_{D_2} = 40$

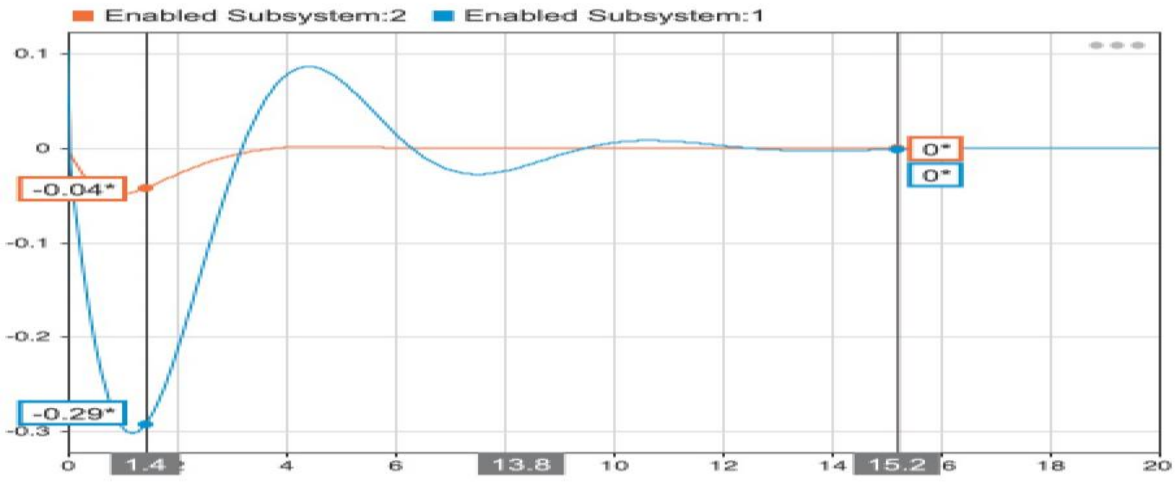
Here we have
1.Peak Value = -0.402 rad, Peak time = 1.124 s, Settling Time = 15.171 s
2.Peak Value = 0.394 rad, Peak time = 0.634 s, Settling Time = 15.171 s

We decrease K_I, because of this settling time is reduced to 15.171. By decreasing K_D, the maximum overshoot value increases by a little amount. As K_P is increased, the number of oscillations decreases.

MATLAB Plot



Simulink Plot

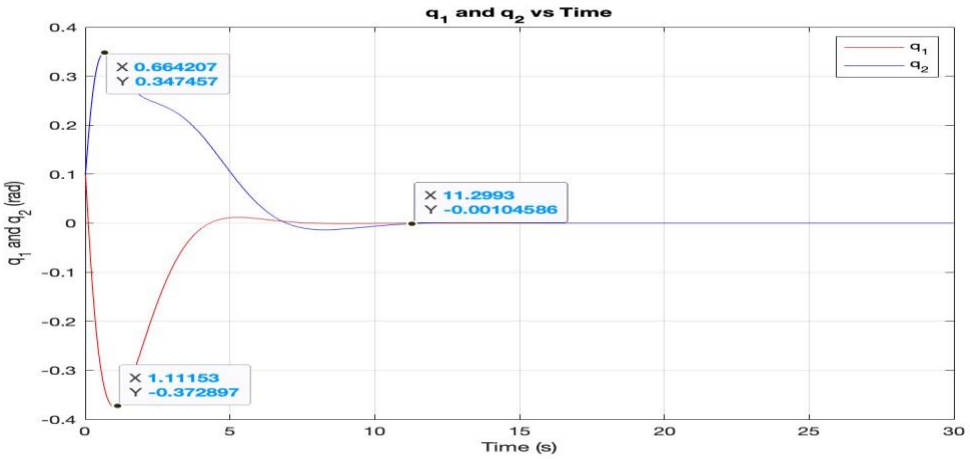


$K_{P_1} = 75, K_{I_1} = 66, K_{D_1} = 50$
$K_{P_2} = 62, K_{I_2} = 40, K_{D_2} = 50$

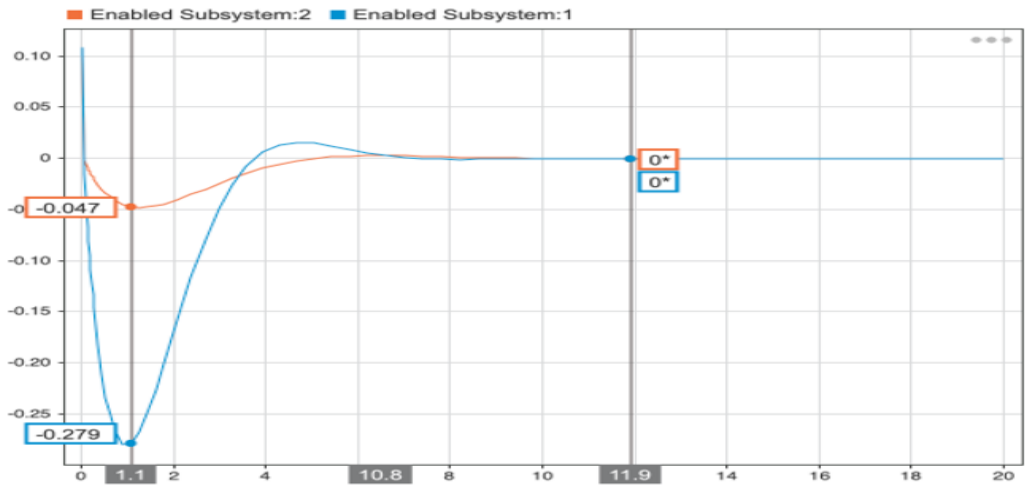
Here we have
1.Peak Value = -0.372 rad, Peak time = 1.111 s, Settling Time = 11.2993 s
2.Peak Value = 0.347 rad, Peak time = 0.664 s, Settling Time = 11.2993 s

In system 2, we kept K_D and K_I, are increased and K_P remains same, because of this Maximum overshoot reduces and Settling time decreases.
In system 1, we increase K_P, K_I and K_D remains same, because of increase in K_P the settling time decreases and maximum overshoot decreases. Because of non-zero value of K_I we get zero steady state error in PID system.

MATLAB Plot



Simulink Plot



Conclusion:

From the above modelling of a 2-link manipulator and taking observations for control action of a PID controller for different values of K_P, K_I and K_D, we can conclude that the basic PID controller can be used very effectively in handling error for two input two output system as with the given system. And with some manual calculations we can control our output based on the desires of the user by just knowing the actions of different components of PID controller and applying them righteously. The proportional control K_P which is responsible for taking the output value close to desired value at a higher speed does not help in reducing the steady state error and oscillation in output. Whereas K_I which causes steady state error to be zero can't help with overshoots and undershoots of the output graph and it causes system to be unstable. K_D which can handle overshooting of graph by keeping a track of rate of change of output is unable to handle steady state error and thus does not provide the user with required result. Hence to get a stable output, which is required, one must take care of all these controllers at once and keep a watch on the shortcoming that comes with these controllers. Keeping track of the shortcomings of each controller along with what they have to offer makes it easier to get the desired output.