Systems Thinking
Mini Project

Team members :

- Yashwanth Duggi
  Roll no :2022102051
- V.V.S.R.Chetan Krishna
  Roll no :2022102047
- T.S.Mitra
  Roll no :2022102061
- Shaik Affan Adeeb
  Roll no :2022102054
- G.S.S Ananya Varma
  Roll no :2022102064
- Vishnu Priya Chekuru
  Roll no :2022102023
- Kushang Agarwal
  Roll no :2022102076
- Sriram Alluri
  Roll no :2022102016

Question :

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

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}+\mathbf{G}(\mathbf{q})=\boldsymbol{\tau},$$
$$\mathbf{M}=\begin{bmatrix} M_{11} & M_{12} \\ M_{12} & M_{22} \end{bmatrix}, \mathbf{q}=\begin{bmatrix} q_1 \\ q_2 \end{bmatrix},$$

$$M_{11}=(m_1+m_2)\,l_1^2+m_2l_2\left(l_2+2l_1\cos\left(q_2\right)\right),$$

$$M_{12}=m_2l_2\left(l_2+l_1\cos\left(q_2\right)\right), M_{22}=m_2l_2^2,$$

$$\mathbf{C}=\begin{bmatrix} -m_2l_1l_2\sin\left(q_2\right)\dot{q}_2 & -m_2l_1l_2\sin\left(q_2\right)\left(\dot{q}_1+\dot{q}_2\right) \\ 0 & m_2l_1l_2\sin\left(q_2\right)\dot{q}_2 \end{bmatrix},$$

$$\mathbf{G}=\begin{bmatrix} m_1l_1g\cos\left(q_1\right)+m_2g\left(l_2\cos\left(q_1+q_2\right)+l_1\cos\left(q_1\right)\right) \\ m_2gl_2\cos\left(q_1+q_2\right) \end{bmatrix}$$

- 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 $[q_1(0)q_2(0)] = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$ rad.

- The objective is to bring the the joint angles from the initial position to $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

- 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

Solution :

2-Link Manipulator: System Dynamics

The system dynamic equation for a 2-link manipulator can be expressed as:

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

Where:

- $M(q)$ is a 2×2 matrix.
- q is a 2×1 matrix vector: $\begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$.
- $C(q,\dot{q})$ is a 2×1 matrix.
- $\dot{q}$ is a 2×1 matrix vector: $\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$.
- G is a 2×1 matrix.

All the matrices are given.
The equation can also be represented as:

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

Now, solving for $\ddot{q}$, we have:

$$\begin{aligned}
&\Rightarrow M(q)\ddot{q} = \tau - [C(q,\dot{q})\dot{q} + G(q)] \\
&\Rightarrow \ddot{q} = M^{-1}(q)[\tau - (C(q,\dot{q})\dot{q} + G(q))] \\
&\Rightarrow \ddot{q} = M^{-1}(q)\tau - M^{-1}(q)[C(q,\dot{q})\dot{q} + G(q)] \longrightarrow [I]
\end{aligned}$$

Assuming:

$$\boxed{\hat{\tau} = M^{-1}(q)\tau} \tag{1}$$

Let :

$$\tau = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

and from equation (1)

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q)\hat{\tau}$$

Denoting Error Signals :
The error signals are denoted as follows:

$$e\left(q_1\right) = q_{1f} - q_1$$
$$e\left(q_2\right) = q_{2f} - q_2$$

Where the target positions of Manipulator Arm 1 and 2 are given by the angles q_{1f} and q_{2f}.
Initial Positions :
The initial positions of the system are given as:

$$q_0 = \begin{bmatrix} q_1(0) \\ q_2(0) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \rightarrow \text{as provided in the question}$$

Modeling PID Group Output :
The PID group output is modeled as:

$$f = k_p e + k_D \dot{e} + k_I \int e dt$$
$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

For f_1:

$$\begin{aligned} f_1 &= k_{p_1} e_1(q_1) + k_{D_1} \dot{e}_1(q_1) + k_{I_1} \int e_1(q_1) dt \\ &= k_{p_1}(q_{1f} - q_1) + k_{D_1}(\dot{q}_{1f} - \dot{q}_1) + k_{I_1} \int (q_{1f} - q_1) dt \\ &= k_{p_1}(q_1 f - q_1) - k_{p_1}\dot{q}_1 + k_{I_1} \int (q_{1f} - q_1) dt \end{aligned}$$

For f_2:

$$\begin{aligned} f_2 &= k_{p_2} e_2(q_2) + k_{D_2} \dot{e}_2(q_2) + k_{I_2} \int e_2(q_2) dt \\ &= k_{p_2}(q_{2f} - q_2) + k_{D_2}(\dot{q}_{2f} - \dot{q}_2) + k_{I_2} \int (q_{2f} - q_2) dt \\ &= k_{p_2}(q_2 f - q_2) - k_{p_2}\dot{q}_2 + k_{I_2} \int (q_{2f} - q_2) dt \end{aligned}$$

Actual Torques for the Plant System :
The actual torques for the plant system are given by:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(\theta) \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Defining State Variables :
The state variables are defined as follows:

$$\begin{aligned} x_1 &= q_1 \\ x_2 &= q_2 \\ x_3 &= \dot{q}_1 \\ x_4 &= \dot{q}_2 \\ x_5 &= \int e_1(q_1)dt \\ x_6 &= \int e_2(q_2)dt \end{aligned}$$

Finding Derivatives of State Variables :
The derivatives of the state variables are given by:

$$\begin{aligned} \dot{x}_1 &= \dot{q}_1 = x_3 \\ \dot{x}_2 &= \dot{q}_2 = x_4 \\ \dot{x}_3 &= \ddot{q}_1 = \phi(x_1, x_2, x_3, x_4, t, x_5, x_6) \\ \dot{x}_4 &= \dot{q}_2 = \psi(x_1, x_2, x_3, x_4, x_5, x_6, t) \\ \dot{x}_5 &= e_1(q_1) = q_1 f - q_1 = q_1 f - x_1 = 0 - x_1 \\ \dot{x}_6 &= e_2(q_2) = q_{2f} - q_2 = q_{2f} - x_2 = 0 - x_2 \end{aligned}$$

Final Desired Angles for 2-link manipulator in our dynamics:
The final desired angles for the robot arms are given as:

$$q_{final} = \begin{bmatrix} q_1 f \\ q_{2f} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

From equation [I]:

$$\begin{aligned} \ddot{q} &= M^{-1}(q)\tau - M^{-1}(q)[c(q, \dot{q})\dot{q} + G(q)] \\ &= \hat{\tau} - M^{-1}(q)[c(q, \dot{q})\dot{q} + G(q)] \\ \ddot{q} &= \begin{bmatrix} \ddot{q}_1 \\ \dot{q}_2 \end{bmatrix} \end{aligned}$$

State Variables and Initial Conditions :

The state variables are represented as:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} \text{ and } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

This first order nonlinear differential equation of the form:

$$\dot{x} = \frac{dx}{dt}, \quad x(0) = \text{ initial conditions for our state variables.}$$

Regarding $x(0)$, considering the initial conditions as:

$$x(0) = \begin{bmatrix} 0.1 \\ 0.1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Initially, the derivative and integral values of q_1 and q_2 are unknown, and since the system is causal, they are taken initially as zero.

ODE45 in MATLAB :

In the code, the 'ode45' command is used to obtain the state variables $\bar{x}$ as output from the differential equation vector $\bar{\dot{x}}$ of the state variables and the vector representing the initial state as the arguments.

$$[t, s] = \text{ode45}(@(t, \text{state})\text{func}(t, \text{state}), t_{\text{span}}, y_0)$$

Where:

- y_0 represents the initial state.
- func(t, state) : Returns the differentials $\dot{x}$ vector.

Main Outputs for Plotting :

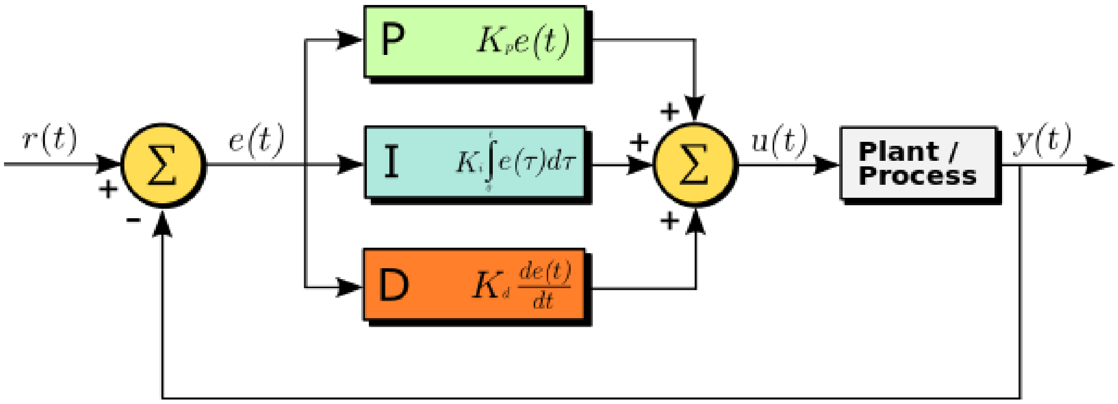
The main outputs to be plotted are q_1 and q_2, which can be obtained using 'ode45'.

$$x(1) = x_1 = q_1$$
$$x(2) = x_2 = q_2$$

These values represent the desired angles for the robot arms.

PID Controller:

A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value $e(t)$ as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.



A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV).

Mathematical form :

The overall control function is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

where K_p, K_i, and K_d (sometimes denoted as P, I, and D) are non-negative coefficients for the proportional, integral, and derivative terms, respectively.

Proportional Term :

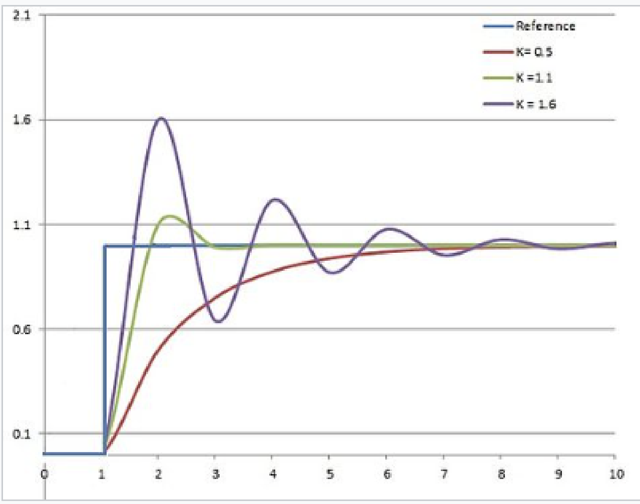
The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain constant.

The proportional term is given by:

$$P_{\mathrm{out}} = K_{\mathrm{p}} \cdot e(t)$$

In this equation:

- P_{out} represents the proportional output.
- K_{p} is the proportional gain constant.
- $e(t)$ represents the current error value.



Response of PV to step change of SP vs time, for three values of K_p (K_i and K_d held constant)

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances.

Integral Term :

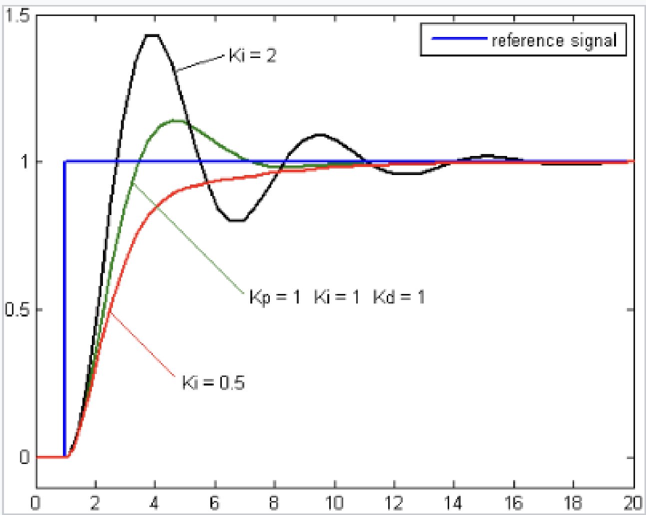
The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. In a PID controller, the integral term is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain K_i and added to the controller output. The integral term is given by:

$$I_{\text{out}} = K_i \int_0^t e(\tau)d\tau$$

Where:

- I_{out} is the integral term output.
- K_i is the integral gain.

This integral term plays a crucial role in PID control systems, helping to eliminate steady-state errors by accumulating and correcting past errors over time.



Response of PV to step change of SP vs time, for three values of K_i (K_p and K_d held constant)

The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value.

Derivative Term

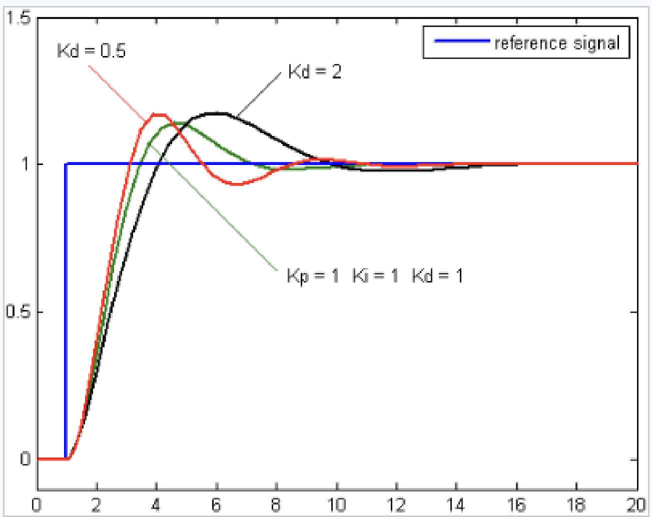
The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d.

The derivative term is given by:

$$D_{\text{out}} = K_d \frac{de(t)}{dt}$$

Where:

- D_{out} is the derivative term output.
- K_d is the derivative gain.



Response of PV to step change of SP vs time, for three values of K_d (K_p and K_i held constant)

Derivative action predicts system behavior and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low-pass filtering for the derivative term to limit the high-frequency gain and noise.

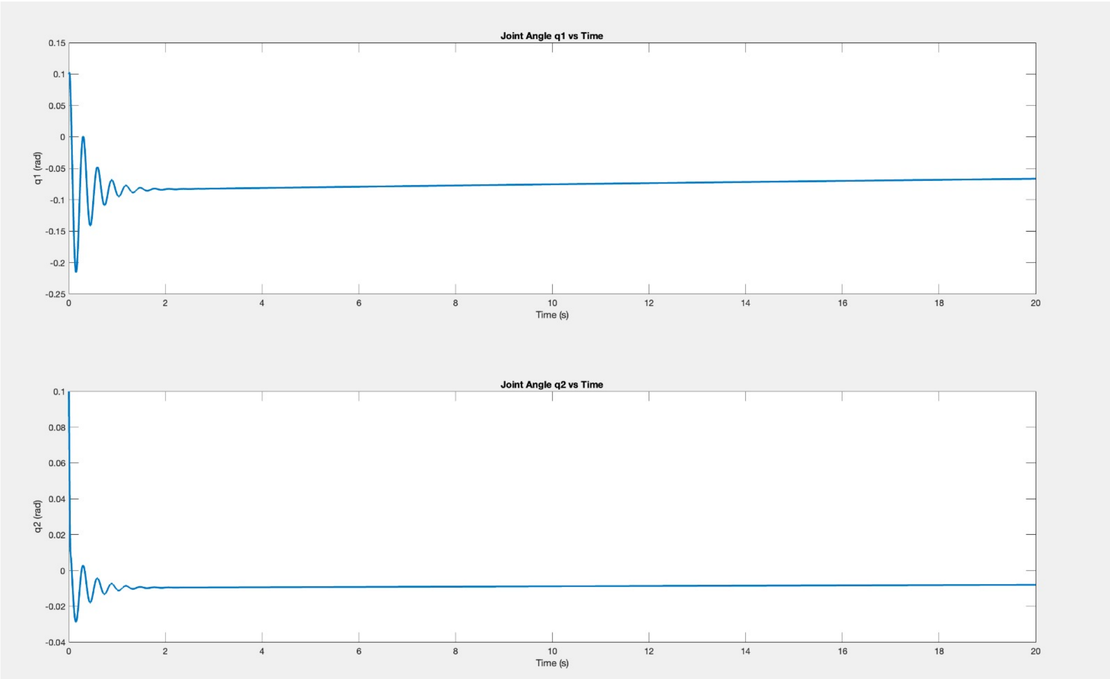
MATLAB Code :

```
% Simulation settings
t_span = 0:0.01:20;
y0 = [0.1, 0.1, 0, 0,0,0]; % Initial joint angles and
    velocities

% Run simulation
[t, s] = ode45(@(t, state) func(t, state), t_span, y0);

% Plotting
figure;
subplot(2, 1, 1);
plot(t, s(:, 1), 'LineWidth', 2);
title('Joint Angle q1 vs Time');
xlabel('Time (s)');
ylabel('q1 (rad)');

subplot(2, 1, 2);
plot(t, s(:, 2), 'LineWidth', 2);
title('Joint Angle q2 vs Time');
xlabel('Time (s)');
ylabel('q2 (rad)');

function der_S = func(t, state)
    % System parameters
    m1 = 10;
    m2 = 5;
    l1 = 0.2;
    l2 = 0.1;
    g = 9.81;

    % Controller gains
    kp1 = 200;
    kd1 = 5;
    kp2 = 400;
    kd2 = 150;
    ki1 = 500;
    ki2 = 600;

    % Extracting state variable
    q1 = state(1);
```

```
    q2 = state(2);
    q1_dot = state(3);
    q2_dot = state(4);
    neg_int_q1 = state(5);
    neg_int_q2 = state(6);

    % Equations of motion
    m11 = (m1 + m2) * (l1^2) + m2 * l2 * (l2 + 2 * l1 *
       cos(q2));
    m12 = m2 * l2 * (l2 + l1 * cos(q2));
    m22 = m2 * (l2^2);

    M = [m11, m12; m12, m22];

    c11 = -m2 * l1 * l2 * sin(q2) * q2_dot;
    c12 = -m2 * l1 * l2 * sin(q2) * (q1_dot + q2_dot);
    c21 = 0;
    c22 = m2 * l1 * l2 * sin(q2) * q1_dot;

    C = [c11, c12; c21, c22];

    g1 = m1 * l1 * g * cos(q1) + m2 * g * (l2 * cos(q1 +
        q2) + l1 * cos(q1));
    g2 = m2 * g * l2 * cos(q1 + q2);

    G = [g1; g2];

    % PDI control law
    tau1 = -kp1 * q1 - kd1 * q1_dot + ki1 * neg_int_q1;
    tau2 = -kp2 * q2 - kd2 * q2_dot + ki2 * neg_int_q2;

    % Control input vector
    Tau = [tau1; tau2];

    % Solve for q1_ddot and q2_ddot
    q_ddot = M \ (Tau - C * [q1_dot; q2_dot] - G);

    % State derivatives
    der_S =[q1_dot;q2_dot;q_ddot(1);q_ddot(2);-q1;-q2];
end
```

Final outputs of joint angles w.r.t time with PID control:

- Case 1 :



- The values of proportional gains are: $K_{p1} = 400$, $K_{p2} = 500$.
- The values of integral gains are: $K_{i1} = 5$, $K_{i2} = 5$.
- The values of derivative gains are: $K_{d1} = 5$, $K_{d2} = 5$.

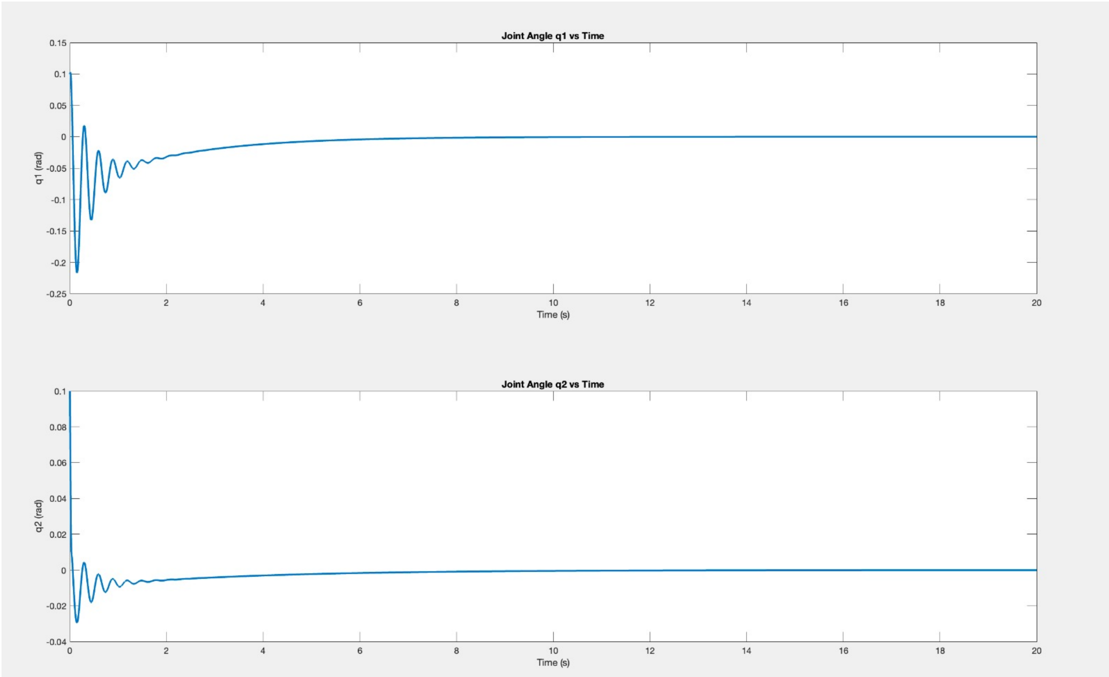
Observations:

1. Oscillatory damping of q_1 and q_2 are observed. The steady-state error values of q_1 and q_2 are non-zero for a timespan of 20 seconds. For example, $e(q_1) = -0.06$ and $e(q_2) = -0.008$ at steady state in the given case.

2. We are able to observe noise which is damped with time.

To overcome these above problems in our observation, we need to tune the gains (K_p, K_d, and K_i) so as to achieve steady state faster, avoid noise, and reduce steady-state error.

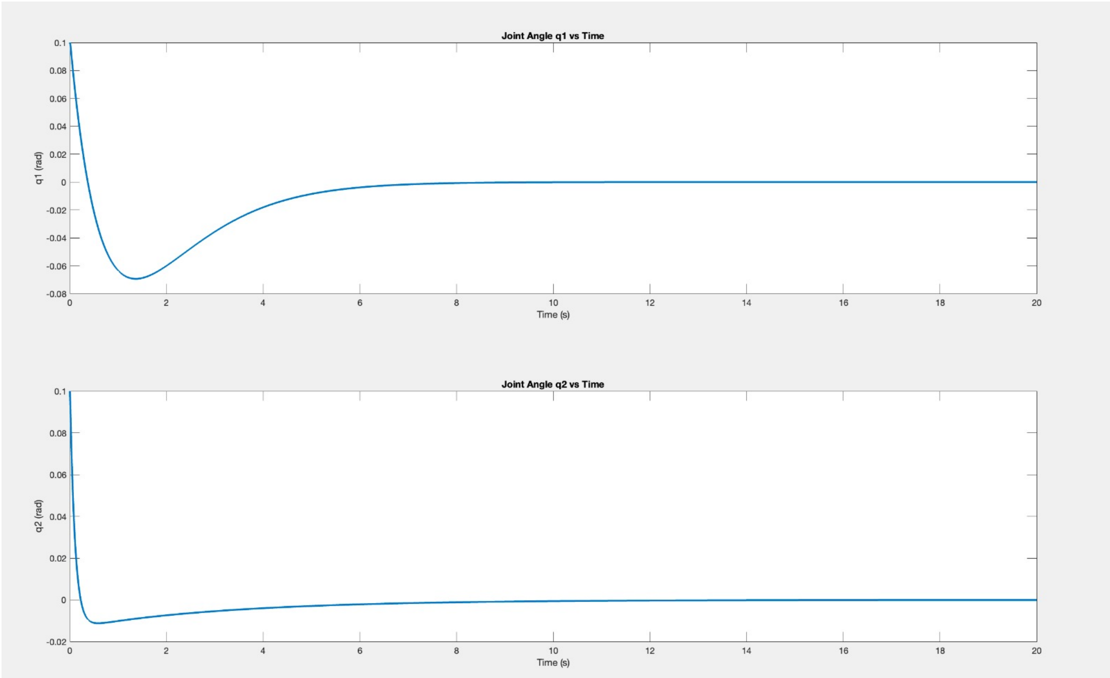
- Case 2 :



- The integral gains K_{i1} and K_{i2} have been increased with the goal of achieving a setpoint of 0 as the system reaches steady state.
- $\Rightarrow e(q_1)$ and $e(q_2)$ approach approximately 0 at steady state.
- The values of integral gains are: $K_{i1} = 200$, $K_{i2} = 150$.
- The values of derivative gains are: $K_{d1} = 5$, $K_{d2} = 5$.
- We observe that the proportional and derivative gains are kept constant, while the integral gains have been increased:
  K_{i1} changed from 5 to 200,
  K_{i2} changed from 5 to 150.
- Since the steady-state error of q_2 was comparatively less than that of q_1, the integral gain of q_1 was increased to a larger extent than that of q_2.

Observation:
Now we observe that the steady-state errors are close to zero. Our aim now is to reduce the noise before achieving steady state, i.e., we need to achieve steady state quickly within our specified time span.

- Case 3 :



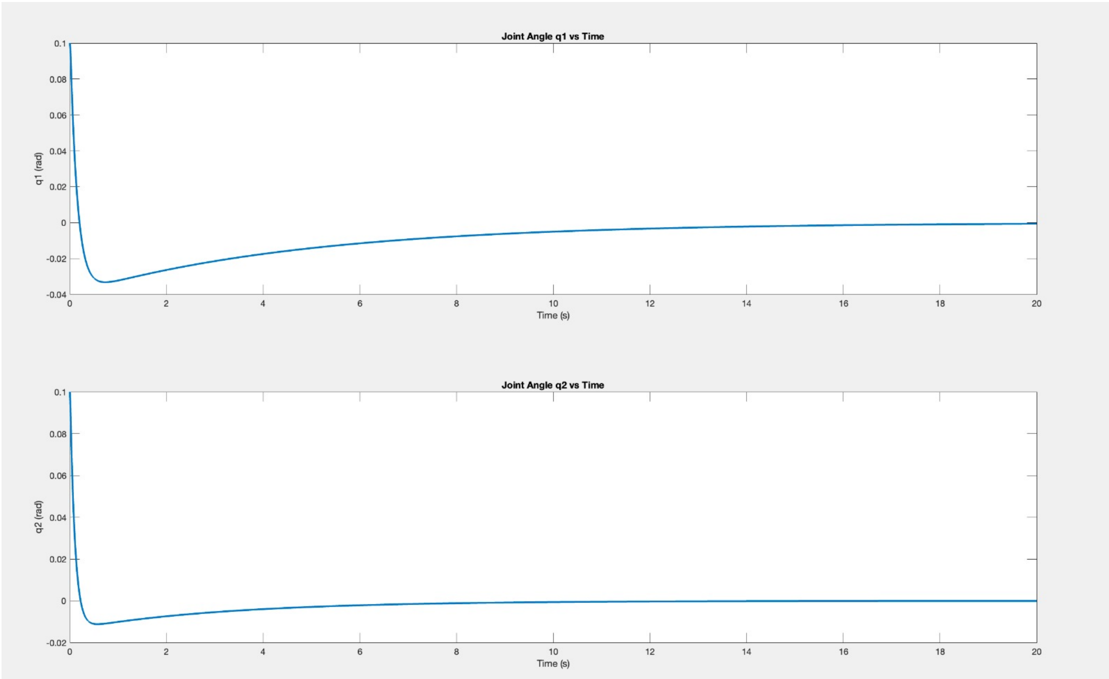
- In order to reduce oscillations, the derivative gains K_{d1} and K_{d2} are increased.
- q_1 exhibits rapid oscillations of larger amplitude compared to q_2. Therefore, K_{d1} is increased to a larger value than K_{d2}.
- The values of proportional gains are: $K_{p1} = 400$, $K_{p2} = 500$.
- The values of integral gains are: $K_{i1} = 200$, $K_{i2} = 150$.
- The values of derivative gains are: $K_{d1} = 200$, $K_{d2} = 50$.
- K_{d1} has been increased from 5 to 200, and K_{d2} has been increased from 5 to 50.

Observations:

The oscillations have reduced, but steady state is achieved in more time for q_1 (approximately 7 seconds) within our time span taken. Our aims are as follows:

1. To make q_1 achieve steady state in less time (leading to case 4).
2. To avoid the initial variation of q_1 about the setpoint when it reaches the setpoint for the first time.

- Case 4 :

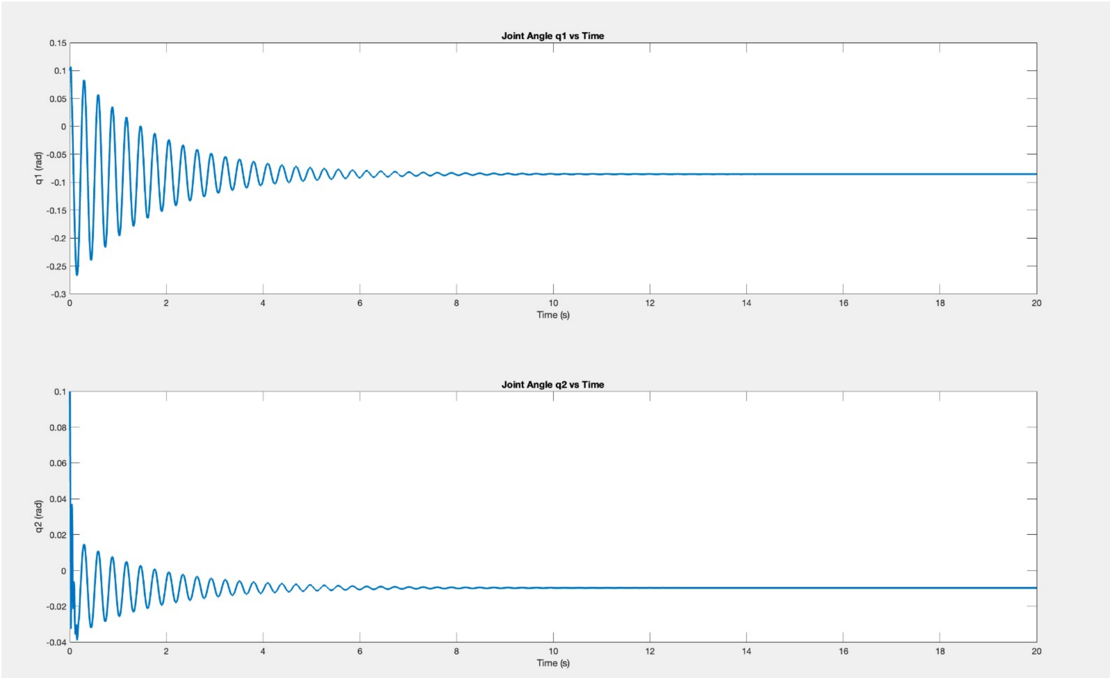


- To further reduce the variation of q_1 about the setpoint and ensure q_1 achieves the setpoint in less time, we have adjusted the controller gains. Specifically, we reduced the derivative gain and increased the proportional gain.
- The values of proportional gains are: $K_{p1} = 1000$, $K_{p2} = 500$.
- The values of integral gains are: $K_{i1} = 200$, $K_{i2} = 150$.
- The values of derivative gains are: $K_{d1} = 150$, $K_{d2} = 50$.
- Changes:
  K_{p1} was increased from 400 to 1000 K_{d1} was reduced from 200 to 150.
- The derivative term generally slows down the rate of change of q_1 with respect to time, which increases the time taken to achieve steady state. Therefore, K_{d1} was reduced to decrease the time taken to achieve steady state. On the other hand, K_{p1} was increased to enhance the rate at which q_1 drops from 0.1 rad to setpoints.
- As a result, the required goal of PID control was achieved, which includes the reduction of oscillations and reaching the given setpoint (0 radians) within the specified time span of 20 seconds.

Final outputs of joint angles w.r.t time with PD control:

In PD controller, we take the integral gain Ki =0

- Case 1 :



- The values of proportional gains are: $K_{p1} = 400$, $K_{p2} = 500$.
- The values of integral gains are: $K_{i1} = 0$, $K_{i2} = 0$.
- The values of derivative gains are: $K_{d1} = 1$, $K_{d2} = 1$.
- Initially, we set our derivative gains as low as possible (here, 1) to demonstrate that q_1 and q_2 exhibit oscillations up to a certain time interval before reaching steady state.

Observations:

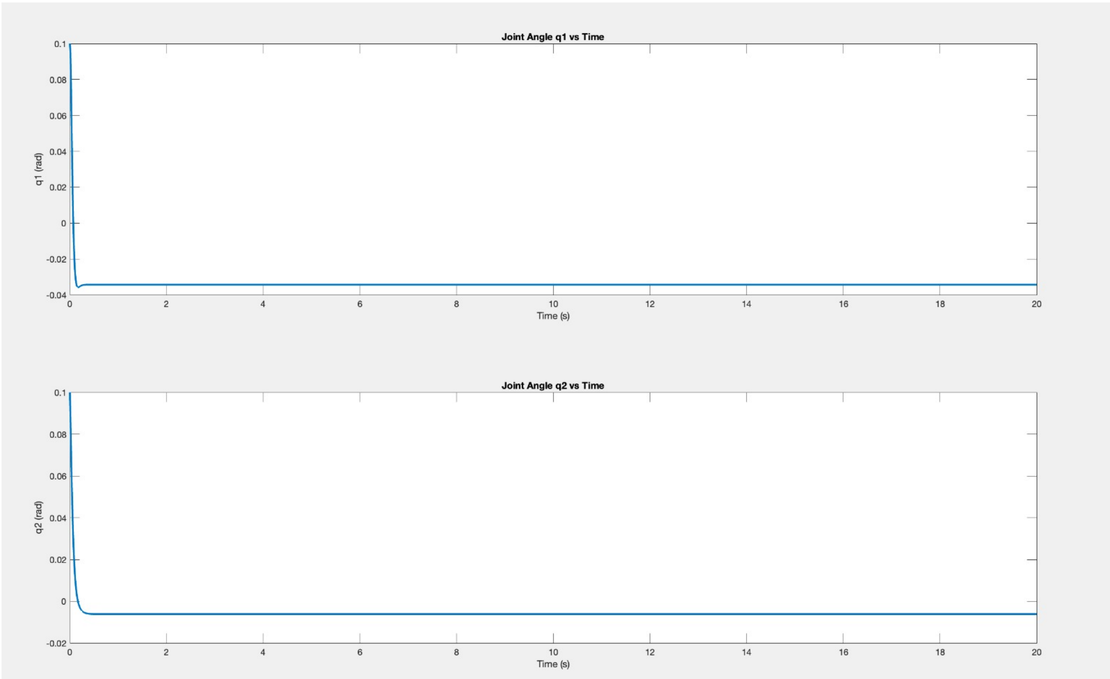
1. Oscillations are observed.

2. The steady-state errors of q_1 and q_2 are non-zero. Typically, the integral term accumulates the error from previous states, multiplied by time, to help the joint angles achieve the setpoint with approximately zero steady-state error.

Aim:

To reduce oscillations and decrease the time taken to achieve steady state.

- **Case 2:**



- The values of proportional gains are: $K_{p1} = 1000$, $K_{p2} = 800$.
- The values of integral gains are: $K_{i1} = 0$, $K_{i2} = 0$.
- The values of derivative gains are: $K_{d1} = 50$, $K_{d2} = 50$.
- To fasten the process and reduce the settling time, the following adjustments were made:
- Proportional gains were increased:
  K_{p1}: $400 \rightarrow 1000$
  K_{p2}: $500 \rightarrow 800$

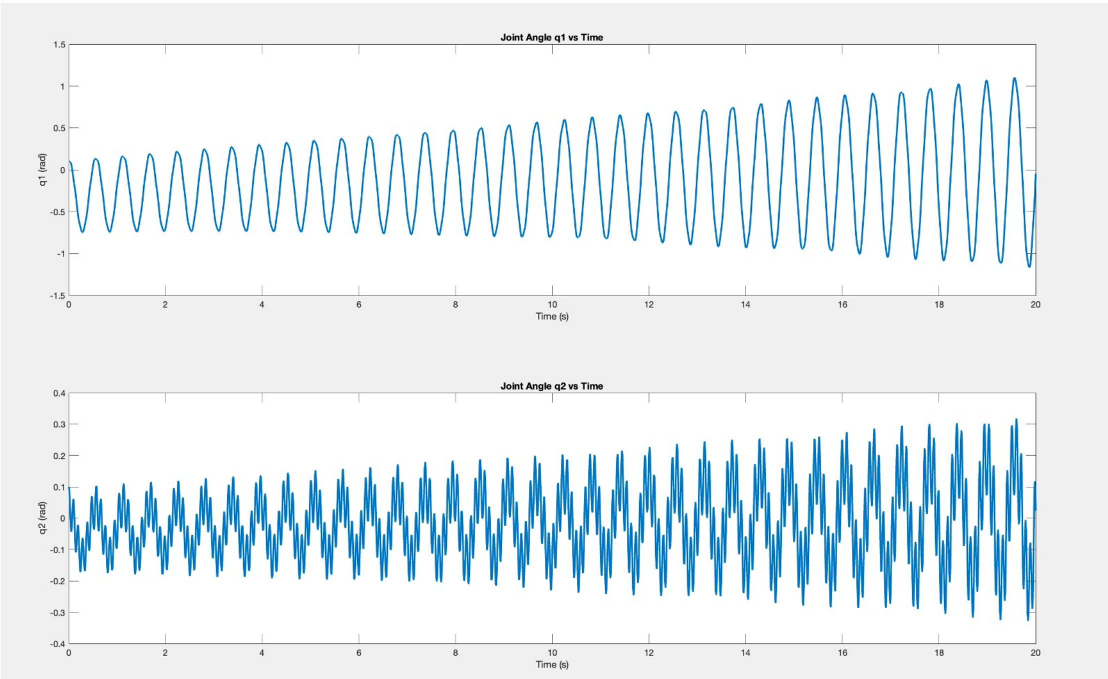
- Derivative gains were increased to reduce oscillations around the setpoint before reaching steady state:
  K_{d1}: $1 \to 50$
  K_{d2}: $1 \to 50$

Observations:
Steady states were achieved with some error (integral gains K_{i1} and K_{i2} set to 0). For these gains, the steady-state errors are approximately:

- $e(q_1) = -0.03$
- $e(q_2) = -0.06$

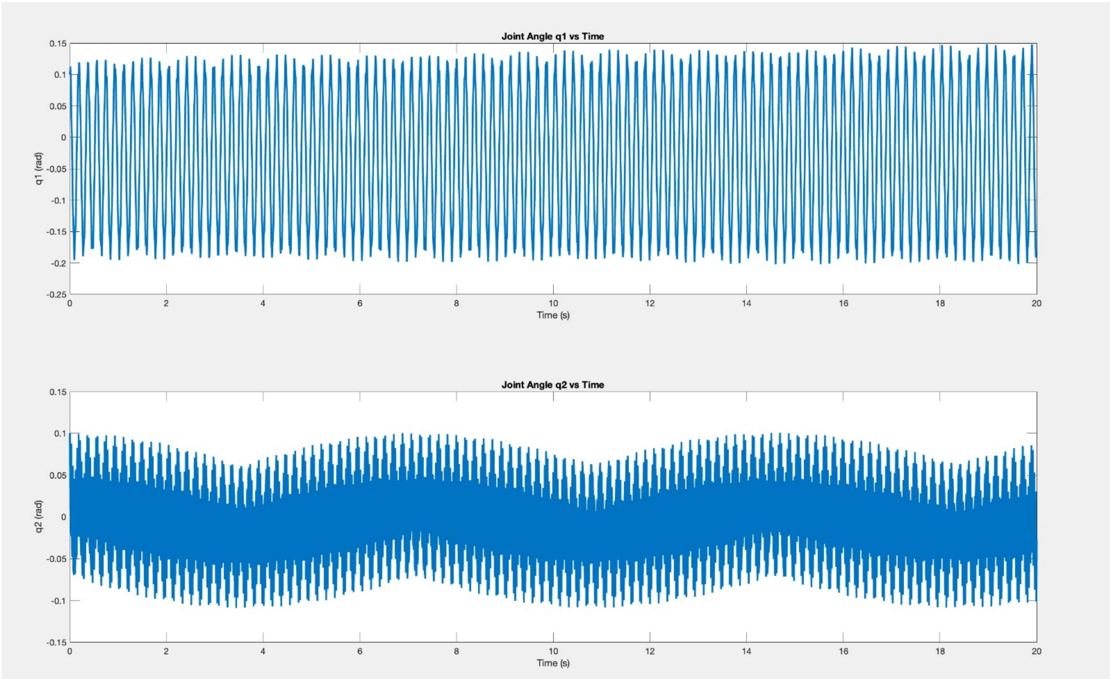
PI Controller :

- Case 1 :



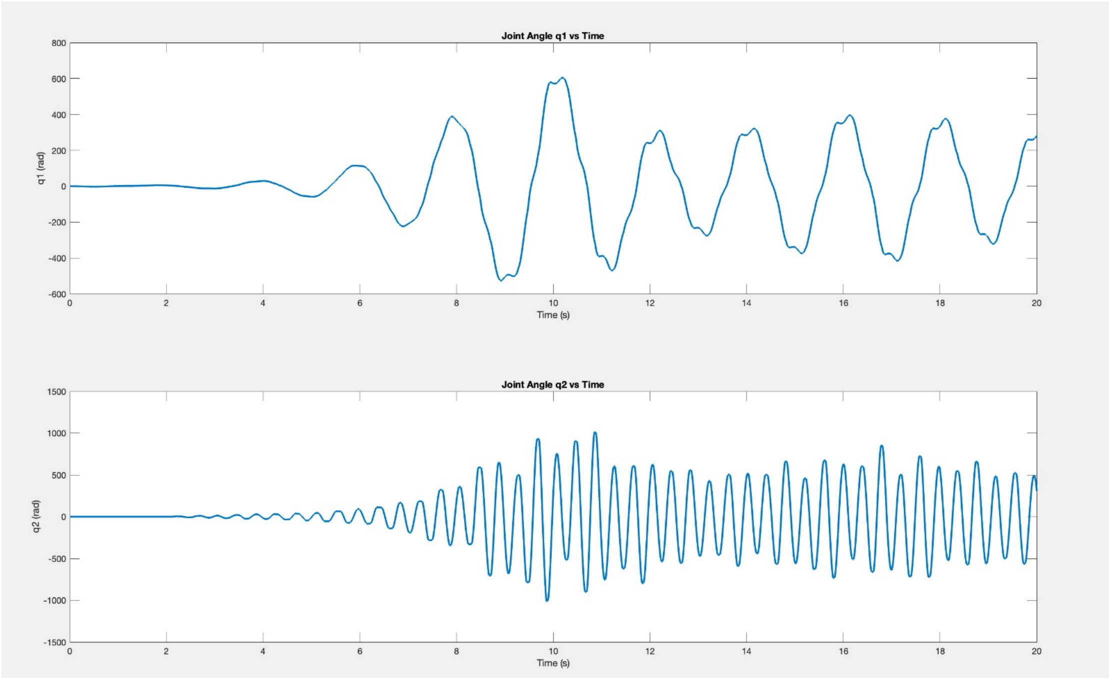
- The values of proportional gains are: $K_{p1} = 100$, $K_{p2} = 100$.
- The values of integral gains are: $K_{i1} = 0$, $K_{i2} = 0$.
- The values of derivative gains are: $K_{d1} = 10$, $K_{d2} = 10$.

- Case 2 :



- The values of proportional gains are: $K_{p1} = 1000$, $K_{p2} = 1000$.
- The values of integral gains are: $K_{i1} = 0$, $K_{i2} = 0$.
- The values of derivative gains are: $K_{d1} = 10$, $K_{d2} = 10$.

- Case 3 :



- The values of proportional gains are: $K_{p1} = 5$, $K_{p2} = 10$.
- The values of integral gains are: $K_{i1} = 0$, $K_{i2} = 0$.
- The values of derivative gains are: $K_{d1} = 10$, $K_{d2} = 10$.

Observations :

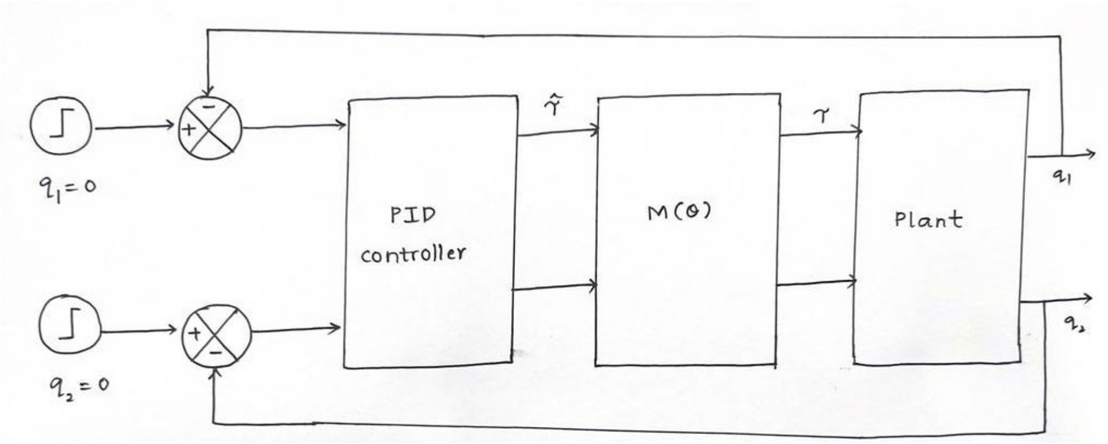
- In all three cases, oscillations are observed for any time span considered, and our joint angles do not reach a steady state.
- In a PI (Proportional-Integral) controller, the integral term accumulates the error over time.
- When the system starts, the integral term accumulates error quickly because there's no damping from the derivative term ($K_d = 0$).
- This can lead to what's called "integral windup," where the integral term becomes very large and causes the system to overshoot and oscillate.

- Without the derivative term, the system can become underdamped, meaning it oscillates around the desired setpoint.

- The proportional and integral terms alone might not provide enough damping to bring the system to a stable state.

Simulink:

To perform the same in simulink we have obtained a block diagram as shown :



To implement a model in Simulink, we have obtained the differential equations for the chosen state variables.

Previously, we demonstrated that $\ddot{q}$ can be derived from the equations:

$$\ddot{q} = M^{-1}(q)\tau - M^{-1}(q)[c(q, \dot{q})\dot{q} + G(q)]$$

The exact equations for $\ddot{q}$ are:

$$M^{-1}(\theta)_{2\times 2}\left[(\dot{\theta} + G)\right]_{2\times 1} + M^{-1}(\theta)\tau$$

$$= \begin{bmatrix} -m_2^2 l_1 l_2^3 \sin x_2 (2x_3x_4 + x_4^2) + m_1 m_2 l_1 l_2^2 g \cos x_1 + m_2^2 g l_2^3 \cos(x_1+x_2) \\ + m_2 l_2^2 l_1 \cos x_1 - m_2^2 l_1 l_2^2 \sin x_2 x_4^2 (l_2 + l_1 \cos x_2) - m_2^2 g l_2^2 \cos(x_1+x_2)(l_2 + l_1 \cos x_3) \\ \\ m_2^2 l_1 l_2^2 \sin x_2 (l_2 + l_1 \cos x_2)(2x_3x_4 + x_4^2) - m_1 m_2 l_1 l_2 g \cos x_1 (l_2 + l_1 \cos x_2) \\ - m_2^2 g l_2^2 \cos(x_1+x_2)(l_2 + l_1 \cos x_2) - m_2^2 g l_1 l_2 \cos x_1 (l_2 + l_1 \cos x_2) \\ + (m_1+m_2) l_1^2 m_2 l_1 l_2 \sin x_2 x_4^2 + (m_1+m_2) l_1^2 m_2 g l_2 \cos(x_1+x_2) \\ + m_2^2 l_1 l_2^2 \sin x_2 x_4^2 (l_2 + 2l_1 \cos x_2) + m_2^2 g l_2^2 \cos(x_1+x_2)(l_2 + 2l_1 \cos x_1) \end{bmatrix}$$

$$= \begin{bmatrix} m_1 m_2 l_1 l_2^2 g \cos x_1 + m_2^2 g l_2^2 l_1 \cos x_1 - m_2^2 g l_2^2 l_1 \cos x_2 \cos(x_1+x_2) - 2m_2^2 l_1 l_2^3 x_3 x_4 \sin x_2 \\ - m_2^2 l_1^2 l_2^2 \sin x_2 x_4^2 \cos x_2 - 2m_2^2 l_1 l_2^3 \sin x_2 x_4^2 \\ \\ m_2^2 g l_1 l_2^2 \cos x_2 \cos(x_1+x_2) + 2m_2^2 l_1 l_2^2 \sin x_2 (l_2 + l_1 \cos x_2) x_4^2 \\ + 2m_2^2 l_1 l_2^2 \sin x_2 x_3 x_4 (l_2 + l_1 \cos x_2) + (m_1+m_2) l_1^2 l_2 m_2 g \cos(x_1+x_2) \\ + (m_1+m_2) l_1^3 m_2 l_2 \sin x_2 x_4^2 - m_2 g l_1 l_2 \cos x_1 (l_2 + l_1 \cos x_2)(m_1+m_2) \end{bmatrix}$$

$$M^{-1}(\theta)\tau$$

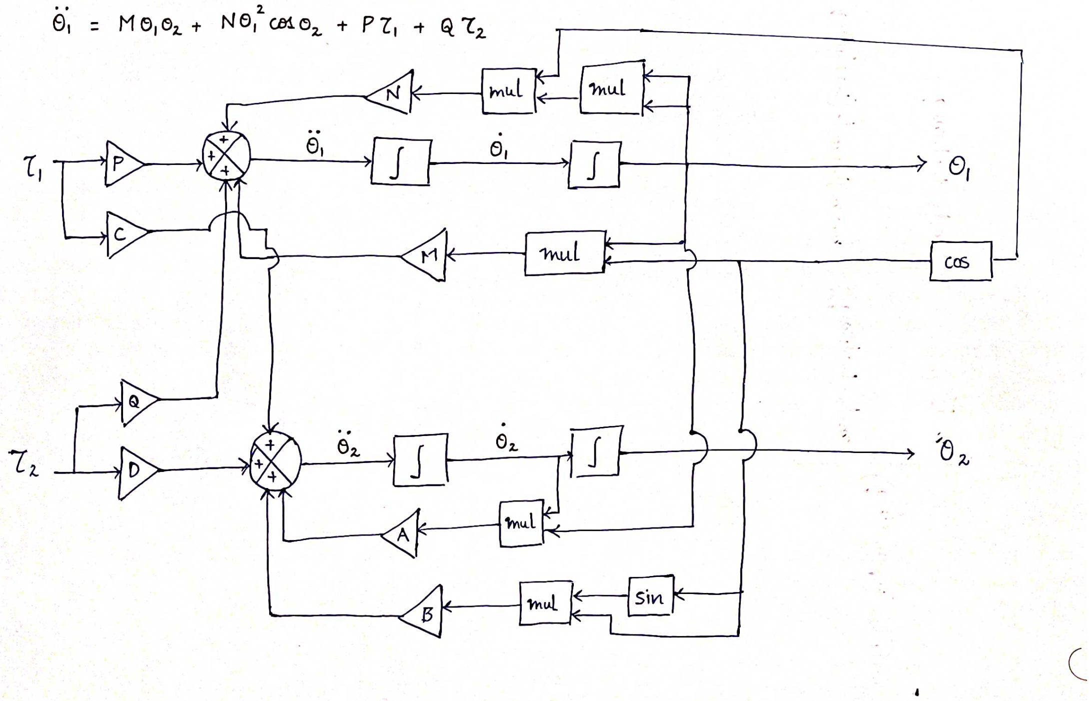
$$= \frac{1}{m_2 L_1^2 L_2^2 (m_1 - m_2 \sin^2 q_2)} \begin{bmatrix} m_2 l_2^2 & -m_2 l_2 (l_2 + l_1 \cos q_2) \\ -m_2 l_2 (l_2 + l_1 \cos q_2) & (m_1 + m_2) l_1^2 + m_2 l_2 (l_2 + 2 l_1 \cos q_2) \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

$$\ddot{q} = M^{-1}(q)\tau - M^{-1}(q)[c(q, \dot{q})\dot{q} + G(q)]$$

The output $\ddot{q}$ from this equation can be used to model the plant in Simulink.

$$\ddot{\theta}_2 = A \theta_1 \dot{\theta}_2 + B \sin\theta_2 (\theta_2) + C \tau_1 + D \tau_2$$

$$\ddot{\theta}_1 = M \theta_1 \theta_2 + N \theta_1^2 \cos\theta_2 + P \tau_1 + Q \tau_2$$



and the plant can be implemented as similar to the above in simulink.

1. Initially, the inputs $r_1(t)$ and $r_2(t)$ (joint angles) are set to 0 rad.

2. The final outputs obtained from the plant are the joint angles obtained for that particular instant, and these q_1 and q_2 are fed back as unity feedback to calculate errors at that instant as shown in the figure.

3. The corresponding errors at that instant are sent into the PID controllers where the controllers are tuned according to our needs.

4. The outputs of the controllers are the angular accelerations $\ddot{q}$, which are further multiplied by the moment of inertia $M(q)$ to obtain the torques, which are the actual inputs to be given to the plant.

5. The plant computes our joint angles from the torques and the state variables taken (q_d and $\ddot{q}$) as shown above.

6. So, finally, our system brings the joint angles to their steady states by estimating the errors in the joint angles at every instant of time and generating the torques from the PID, PD, PI controllers tuned respectively.