System Thinking Project

Team Name: Chitti 8.O

Teammates

Sanjana Sheela
2023102027

Khyathi
2023102065

Keerthi Seela
2023102012

Sahithi Anumula
2023112002

Snigdha STP
2023102036

Kaamya Dasika
2023102034

Pragnya T
2023102067

Gandlur Valli
2023102068

Problem Statement:

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

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

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}, q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$M_{11} = (m1 + m2)l_1^2 + m_2 l_2 (l_2 + 2l_1 cos(q_2)),$$

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

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

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

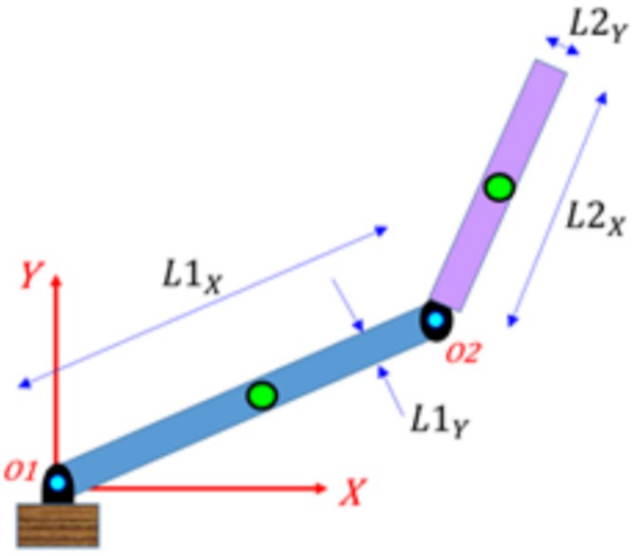
where (m_1, l_1, q_1) and (m_2, l_2, q_2) denote the mass, length and joint angle positions of link 1 and 2 respectively.

The following parametric values are selected: $m_1 = 10kg, m_2 = 5kg, l_1 = 0.2m, l_2 = 0.1m, g = 9.81m/s^2$

The joint angles are initially at positions
$\begin{bmatrix} q_1(0) & q_2(0) \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$ rad.

Objective: To bring the joint angles from the initial position to $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

Dynamics of Two-Link Manipulator



The two-link manipulator consists of two masses connected by weightless bars, with each mass acting as simple pendulum. The system has two degrees of freedom, represented by angles θ_1 *and* θ_2 which represents he angle in which the first bar rotates about the origin and second bar rotates about the endpoint of the first bar respectively.

The Euler - Lagrange equation is as follows:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}\right] - \frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} = \tau_i \qquad (1.1)$$

where i = 1,2 where τ_i is the torque applied to the i^{th} link.

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

$$x_1 = L_1 cos(\theta_1) \quad y_1 = L_1 sin(\theta_1) \qquad (1.2)$$

$$x_2 = L_1 cos(\theta_1) + L_2 cos(\theta_2) \quad y_2 = L_1 sin(\theta_1) + L_2 sin(\theta_2) \qquad (1.3)$$

$$v_1 = \sqrt{\dot{x}_1^2 + \dot{y}_1^2} \quad v_2 = \sqrt{\dot{x}_2^2 + \dot{y}_2^2} \qquad (1.4)$$

$$\dot{x}_1 = -L_1 \dot{\theta}_1 sin(\theta_1) \quad \dot{y}_1 = L_1 \dot{\theta}_1 cos(\theta_1) \qquad (1.5)$$

$$\dot{x}_2 = -L_1 \dot{\theta}_1 sin(\theta_1) - L_2 \dot{\theta}_2 sin(\theta_2) \qquad \dot{y}_2 = L_1 \dot{\theta}_1 cos(\theta_1) + L_2 \dot{\theta}_2 cos(\theta_2) \qquad (1.6)$$

where $\dot{\theta}_k = \frac{d\theta_k}{dt} \;\; and \;\; \dot{x}_k = \frac{dx_k}{dt}$ for k = 1,2 the dot is a derivative with respect to t

The kinetic energy can be written as

$$KE = \frac{1}{2} M_1 v_1^2 \;\; + \;\; \frac{1}{2} M_2 v_2^2 \qquad (1.7)$$

in this if we substitute the expression of v_1 and v_2 from equation 1.1

$$KE = \frac{1}{2} M_1 (\dot{x}_1^2 + \dot{y}_1^2)^2 \;\; + \;\; \frac{1}{2} M_2 (\dot{x}_2^2 + \dot{y}_2^2)^2 \qquad (1.8)$$

By substituting the expressions of (1.2), (1.3) into expression (1.8) we get

$$KE = \frac{1}{2} M_1 ((-L_1 \dot{\theta}_1 sin(\theta_1))^2 + (L_1 \dot{\theta}_1 cos(\theta_1))^2)^2 \;\; + \;\; \frac{1}{2} M_2 ((-L_1 \dot{\theta}_1 sin(\theta_1) - L_2 \dot{\theta}_2 sin(\theta_2))^2 + (L_1 \dot{\theta}_1 cos(\theta_1) + L_2$$

after simplifying 1.9 equation we get

$$KE = \frac{1}{2}(M_1 + M_2) L_1^2 \dot{\theta}_1^2 + \frac{1}{2} M_2 L_2^2 \dot{\theta}^2 + M_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 cos(\theta_1 - \theta_2) \qquad (1.10)$$

The potential energy of the system due to gravity of the i^{th} pendulum is

$$PE_i(\theta) = M_i g h_i(\theta) \quad where \quad i = 1, 2 \qquad (1.11)$$

where h_i is the height of the center of mass of the i^{th} pendulum, g is the acceleration due to gravity constant and M_i is the mass of the i^{th} pendulum.

$$PE(\theta) = \sum_{i=1}^{2} PE_i(\theta) = \sum_{i=1}^{2} M_i g h_i(\theta) = M_1 g L_1 sin(\theta_1) + M_2 g (L_1 sin(\theta_1) + L_2 sin(\theta_2)) = (M_1 + M_2) g L_1 sin(\theta_1) -$$

By substituting the expressions of the (1.10) and (1.12) we get

$$\mathcal{L} = \frac{1}{2}(M_1 + M_2) L_1^2 \dot{\theta}_1^2 + \frac{1}{2} M_2 L_2^2 \dot{\theta}^2 + M_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 cos(\theta_1 - \theta_2) \; - \sum_{i=1}^{2} PE_i(\theta) = \sum_{i=1}^{2} M_i g h_i(\theta) = M_1 g L_1 sin(\theta_1) -$$

By finding the partial derivative the equation 1.13 and simplifying the equation we get

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (M_1 + M_2) L_1^2 \dot{\theta}_1 + M_2 L_1 L_2 \dot{\theta}_2 cos(\theta_1 - \theta_2) \qquad (1.14)$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -M_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 sin(\theta_1 - \theta_2) - (M_1 + M_2) g L_1 cos(\theta_1) \qquad (1.15)$$

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right] = (M_1+M_2)L_1^2\ddot{\theta}_1 + M_2L_1L_2\ddot{\theta}_2 cos(\theta_1-\theta_2) - M_2L_1L_2\dot{\theta}_2(\dot{\theta}_1-\dot{\theta}_1)sin(\theta_1-\theta_2) \tag{1.16}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = M_2L_2\dot{\theta}_2 + M_2L_1L_2\dot{\theta}_1 cos(\theta_1-\theta_2) \tag{1.17}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = M_2L_1L_2\dot{\theta}_1\dot{\theta}_2 sin(\theta_1-\theta_2) - M_2gL_2cos(\theta_2) \tag{1.18}$$

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right] = M_2L_2^2\ddot{\theta}_2 + M_2L_1L_2\ddot{\theta}_1 cos(\theta_1-\theta_2) - M_2L_1L_2\dot{\theta}_1(\dot{\theta}_1-\dot{\theta}_2)sin(\theta_1-\theta_2) \tag{1.19}$$

Using the following equation and the above expressions (1.14) ,(1.16) and (1.19),(1.17)

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right] - \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = \tau_1$$

$$(M_1+M_2)L_1^2\ddot{\theta}_1 + M_2L_1L_2\ddot{\theta}_2 cos(\theta_1-\theta_2) + M_2L_1L_2\dot{\theta}_2^2 sin(\theta_1-\theta_2) + (M_1+M_2)gL_1cos(\theta_1) = \tau_1 \tag{1.20}$$

by changing the equation slightly we get the following equation

$$L_1\ddot{\theta}_1 + \delta L_1L_2\ddot{\theta}_2 cos(\theta_1-\theta_2) = \frac{\delta\tau_1}{M_2L_1} - \delta L_2\dot{\theta}_2^2 sin(\theta_1-\theta_2) - gcos(\theta_1)$$

where $\delta = \frac{M_2}{M_1+M_2}$

doing the same for the second degree of freedom.

$$M_2L_2^2\ddot{\theta}_2 + M_2L_1L_2\ddot{\theta}_2 cos(\theta_1-\theta_2) - M_2L_1L_2\dot{\theta}_1^2 sin(\theta_1-\theta_2) + M_2hL_2cos(\theta_2) = \tau_2$$

By changing the equation slightly we get the following equation

$$L_2\ddot{\theta}_2 + L_1\ddot{\theta}_1 cos(\theta_1-\theta_2) = \frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^2 sin(\theta_1-\theta_2) - gcos(\theta_2)$$

$$g_1 = \frac{\frac{\tau_1\delta}{M_2L_1} - \delta L_2\dot{\theta}_2^2 sin(\theta_1-\theta_2) - gcos(\theta_1) - \delta cos(\theta_1-\theta_2)(\frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^2 sin(\theta_1-\theta_2) - gcos(\theta_2))}{L_1(1-\delta cos^2(\theta_1-\theta_2)}$$

$$g_2 = \frac{\frac{\tau_2}{M_2L_2} - \delta L_1\dot{\theta}_1^2 sin(\theta_1-\theta_2) - gcos(\theta_2) - \delta cos(\theta_1-\theta_2)(\frac{\delta\tau_1}{M_2L_1} + L_2\dot{\theta}_2^2 sin(\theta_1-\theta_2) - gcos(\theta_1))}{L_2(1-\delta cos^2(\theta_1-\theta_2)}$$

we solve the equation for the angles θ_1 and θ_2 for that we need to solve the second order system of ordinary differential equations

We will reduced the system into an equivalent system into an equivalent system of first-order ordinary differential equations.

Four new variables:

$$u_1 = \theta_1, u_2 = \dot{\theta}_1, u_3 = \theta_2, u_4 = \dot{\theta}_2$$

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

We get this first order nonlinear differential equations of the form

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

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

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

$$u_1(0) = \theta_1(0), u_2(0) = \theta_2(0), u_3(0) = \dot{\theta}_1(0), u_4(0) = \dot{\theta}_2(0)$$

The system subject to the initial condition can be solved for the vector U, we will solve the equation in the MATLAB -the ordinary differential equations.

Solution for the 2-Link Manipulator System:

Model Description of the Robotic Manipulator:

The system 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.

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

- q is a 2 × 1 matrix vector:

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

- C($q, \dot{q}$) is a 2 × 1 matrix.

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

- $\dot{q}$ is a 2 × 1 matrix vector:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

- G is a 2 × 1 matrix.

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

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

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

$$\ddot{q} = \tau M^{-1}(q) - M^{-1}(q)\left[G(q) + C(q,\dot{q})\dot{q}\right] \qquad \rightarrow [1]$$

Assuming:

$$\hat{\tau} = M^{-1}(q)\tau \qquad \rightarrow [I]$$

Let,

$$\tau = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

and from equation[I]

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

Initial Positions :

The initial positions of the system are given as:

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

Denoting Error Signals :

The error signals are denoted as follows:

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

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

where the target positions of Manipulator Arm 1 and 2 are given by the angles q_{1f} and q_{2f}.

Modeling PID Group Output :

The PID group output is modeled as:

$$r = k_p e + k_D \dot{e} + k_I \int e \, dt$$

$$r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

Actual Torques for the System:

The actual torques for the plant system are given by:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

State Variables of the System :

The state variables are defined as follows:

$$x_1 = q_1$$

$$x_2 = q_2$$

$$x_3 = \dot{q}_1$$

$$x_4 = \dot{q}_2$$

$$x_5 = \int e_1(q_1) dt$$

$$x_6 = \int e_2(q_2) dt$$

Derivatives of State Variables for the State Equation:

The derivatives of the state variables are given by:

$$\dot{x}_1 = \dot{q}_1 = x_3$$

$$\dot{x}_2 = \dot{q}_2 = x_4$$

$$\dot{x}_3 = \ddot{q}_1$$

$$\dot{x}_4 = \dot{q}_2$$

$$\dot{x}_5 = e_1(q_1) = q_{1f} - q_1 = q_{1f} - x_1 = 0 - x_1$$

$$\dot{x}_6 = e_2(q_2) = q_{2f} - q_2 = q_{2f} - x_2 = 0 - x_2$$

For r_1:

$$r_1 = k_{p1}e_1(q_1) + k_{D1}\dot{e}_1(q_1) + k_{I1}\int e_1(q_1)\,dt$$

$$= k_{p1}(q_{1f} - q_1) + k_{D1}(\dot{q}_{1f} - \dot{q}_1) + k_{I1}\int (q_{1f} - q_1)dt$$

$$= k_{p1}(q_{1f} - q_1) - k_{p1}\dot{q}1 + k_{I1}\int (q_{1f} - q_1)dt$$

For r_2:

$$r_2 = k_{p2}e_2(q_2) + k_{D2}\dot{e}_2(q_2) + k_{I_2}\int e_2(q_2)dt$$

$$= k_{p2}(q_{2f} - q_2) + k_{D2}(\dot{q}_{2f} - \dot{q}_2) + k_{I_2}\int (q_{2f} - q_2)dt$$

$$= k_{p2}(q_{2f} - q_2) - k_{p2}\dot{q}_2 + k_{I2}\int (q_{2f} - q_2)dt$$

Final Desired Angles for 2-link Manipulator in our Dynamics:

The final desired angles for the robot arms are given as:

$$q_{final} = \begin{bmatrix} q_{1f} \\ q_{2f} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

$$= \hat{\tau} - M^{-1}(q)\left[G(q) + C(q,\dot{q})\dot{q}\right]$$

$$\ddot{q} = \begin{bmatrix} \ddot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

State Variables and Initial Conditions:

The state variables are represented as:

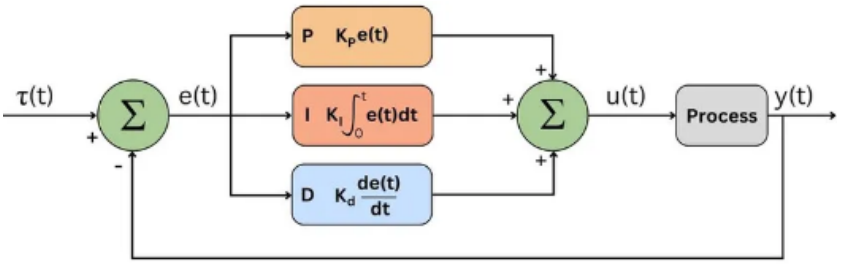
$\dot{x} = \frac{dx}{dt}, x(0)$ = initial conditions for our state variables.

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

Initially, the derivative and integral values of q1 and q2 are unknown, and since the system is causal, they are taken initially as zero.

PID controller:

A Proportional-Integral-Derivative controller is an instrument which receives input data, from sensors and calculates the difference between the actual value and the desired setpoint, and adjusts outputs to control variables such as temperature, speed, voltage etc. It does this through three mechanisms: proportional control, which reacts to current error; integral control, which addresses accumulated past errors; and derivative control, which predicts future errors. This architecture allows PID controllers to efficiently maintain process control and system stability.



The overall control function is given by

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

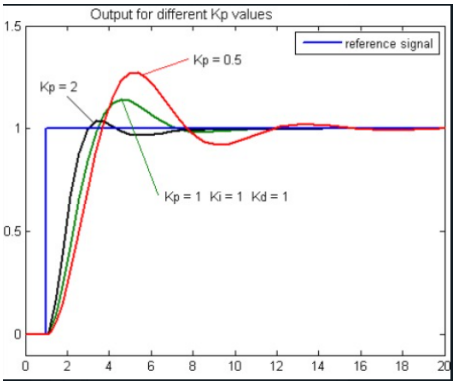
where K_p, K_i, and K_d (sometimes denoted as P, I, and D) are non-negative coefficients for the proportional, integral, and derivative terms, respectively.

Proportional term:

The proportional component gives an output which is proportional to the current error. The proportional gain (K_p) determines the ratio of output response to the error signal. The proportional term is given by:

$$P_{out} = K_p e(t)$$

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



In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If K_p is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller.

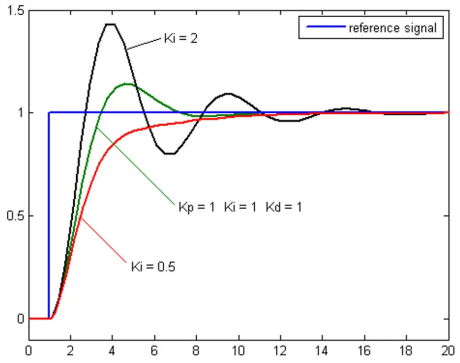
Integral term :

The integral component sums the error term over time. 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. This accumulated error is then multiplied by K_i, the integral constant

and added to the controller output.

$$I_{out} = K_i \int_0^t e(t)dt$$

- I_{out} represents the integral output
- K_i is the integral gain



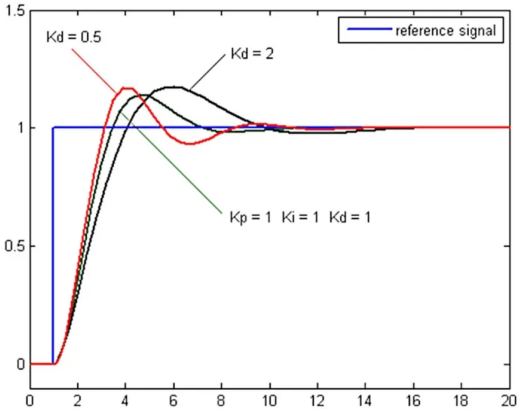
As the integral term accumulates and corrects the past errors, it eliminates the steady state errors. The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral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

Derivative term:

The derivative response is proportional to the rate of change of error. The derivative term is calculated by multiplying the slope of the error over time and a constant (derivative gain) K_d.

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

- D_{out} represents the derivative output
- K_d is the derivative gain



Derivative action predicts system behavior and thus improves settling time and stability of the system. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable.

MATLAB:

```
clear; clc;
% Initial joint angular velocities (q_dot) for both joints
qdot_initial = [0; 0];
time_span = [0 10];

q_initial = [0.1; 0.1];
% Initial integral errors for both joints (for use in integral control)
x_initial = [0; 0];

qRequired = [0; 0];
```

```
% Simulating the system using ode45
[t, states] = ode45(@dynamics_manipulator, time_span, [q_initial; qdot_initial; x_initial]);

%joint angles
q1 = states(:,1);
q2 = states(:,2);

x1 = states(:,5);
x2 = states(:,6);

q1_dot = states(:,3);
q2_dot = states(:,4);



e1 = qRequired(1) - q1;
e2 = qRequired(2) - q2;
% for title
Kp = 250; Kd = 10; Ki = 0;

% Plot q1 and q2 versus time
figure;
subplot(2,1,1);
plot(t, q1);
title("q1");
xlabel("Time (s)");
ylabel("q1(t)");

% error
subplot(2,1,2);
plot(t, q2);
title("q2");
xlabel("Time (s)");
ylabel("q2(t)");

sgtitle("q1, q2 vs t for Kp = " + Kp + ", Kd = " + Kd);


figure; subplot(2,1,1);
plot(t,e1,'r');
title("Error in q1");
xlabel("Time(s)");
ylabel("e_1(t)");
subplot(2,1,2);
plot(t,e2,'r');
title("Error in q2");
xlabel("Time(s)");
ylabel("e_2(t)");

sgtitle("Error plots for Kp = " + Kp + ", Kd = " + Kd);

function dx_dt = dynamics_manipulator(t, states)
% Extract states from input (joint angles, velocities, and integral errors)
    q1_dot = states(3);
    q2_dot = states(4);

    x1 = states(5);
```

```
    x2 = states(6);

    q1 = states(1);
    q2 = states(2);

    % PID controller gains for each joint (same for q1 and q2)
    Kp1 = 250; Kd1 = 10; Ki1 = 0;
    Kp2 = 250; Kd2 = 10; Ki2 = 0;

    q_desired = [0; 0];

    e1 = q_desired(1) - q1;
    e2 = q_desired(2) - q2;

    f1 = Kp1*e1 + Ki1*x1 - Kd1*q1_dot;
    f2 = Kp2*e2 + Ki2*x2 - Kd2*q2_dot;
    F = [f1; f2];

    m1 = 10; m2 = 5;
    l1 = 0.2; l2 = 0.1;
 % Inertia matrix for the two-link manipulator
    M11 = (m1+m2)*(l1^2) + m2*l2*(l2+2*l1*cos(q2));
    M22 = m2*(l2^2);
    M12 = m2*l2*(l2+l1*cos(q2));

    M = [M11, M12; M12, M22];

    %the joint torques based on the control inputs and inertia matrix
    tau = M*F;
    tau1 = tau(1);
    tau2 = tau(2);

    %Dynamics of the system
    dx_dt = zeros(size(states));
    dx_dt(1) = q1_dot;
    dx_dt(2) = q2_dot;

    dx_dt(3) = - (5*(tau1 - (981*cos(q1 + q2))/200 - (2943*cos(q1))/100 + (q1_dot*q2_dot*sin(
    dx_dt(4) = (5*(2*cos(q2) + 1)*(tau1 - (981*cos(q1 + q2))/200 - (2943*cos(q1))/100 + (q1_d

    dx_dt(6) = q_desired(2) - q2;
    dx_dt(5) = q_desired(1) - q1;

end
```

```
m1 = 10;
m2 = 5;
g = 9.81;
l1 = 0.2;
l2 = 0.1;


syms q1; syms q2;
syms q1_dot; syms q2_dot;
syms tau1; syms tau2;

q = [q1; q2];
```

```
syms x1; syms x2;
tau = [tau1; tau2];

q_dot = [q1_dot; q2_dot];


M11 = (m1+m2)*(l1^2) + m2*l2*(l2+2*l1*cos(q2));
M12 = m2*l2*(l2+l1*cos(q2));
M22 = m2*(l2^2);

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)*q2_dot;

G1 = m1*l1*g*cos(q1)+m2*g*(l2*cos(q1+q2)+l1*cos(q1));
G2 = m2*g*l2*cos(q1+q2);

M = [M11, M12;
    M12, M22];


G = [G1; G2];
C = [C11, C12;
    C21, C22];

% tau = M*q_dd + C*q_dot + G;
q_dd = (M^(-1))*(tau - C*q_dot - G);
q1_dd = q_dd(1);
q2_dd = q_dd(2);
disp(q_dd);

qRequired = [0; 0];

e1 = qRequired(1) - q1;
e2 = qRequired(2) - q2;

% To calculate tau1 & tau2 if control variables are given

syms Kp1; syms Kp2;
syms Kd1; syms Kd2;
syms Ki1; syms Ki2;

f2 = Kp2*e2 + Ki2*x2 - Kd2*q2_dot;
f1 = Kp1*e1 + Ki1*x1 - Kd1*q1_dot;

F = [f1; f2];

tau = M*F;
tau1 = tau(1);
tau2 = tau(2);

% disp(f1);

% qdd =
% [- (5*(tau1 - (981*cos(q1 + q2))/200 - (2943*cos(q1))/100 +
% (q1_dot*q2_dot*sin(q2))/10 + (q2_dot*sin(q2)*(q1_dot +
% q2_dot))/10))/(cos(q2)^2 - 3) - (5*(2*cos(q2) + 1)*((sin(q2)*q2_dot^2)/10 - tau2 + (981*cos
```

```
% (5*(2*cos(q2) + 1)*(tau1 - (981*cos(q1 + q2))/200 - (2943*cos(q1))/100 +
% (q1_dot*q2_dot*sin(q2))/10 + (q2_dot*sin(q2)*(q1_dot + q2_dot))/10))/(cos(q2)^2 - 3) + (5*(
```

Explanation for the code:

t_span = [0 10], this sets the time span for the simulation from 0 to 10 seconds.
q_initial, qdot_initial, and x_initial define the initial joint angles, joint velocities, and control variables for the simulation.
x denotes the integral of error, which is the difference in the final equilibrium state and the current angle.

q_desired represents the desired joint angles that the controller aims to achieve. The final q_desired value should be 0 for both q_1 and q_2.

ode45 is a MATLAB function for solving ordinary differential equations (ODEs) using the specified function compute_dynamics.It integrates the dynamics of the manipulator over time, starting from the initial conditions provided. The results are stored in t(time) and states(joint angles, joint velocities, and control variables).

Kp1, Kd1, and Ki1 represent the proportional, derivative, and integral gains for the first joint (q1).
Kp2, Kd2, and Ki2 represent the proportional, derivative, and integral gains for the second joint (q2).

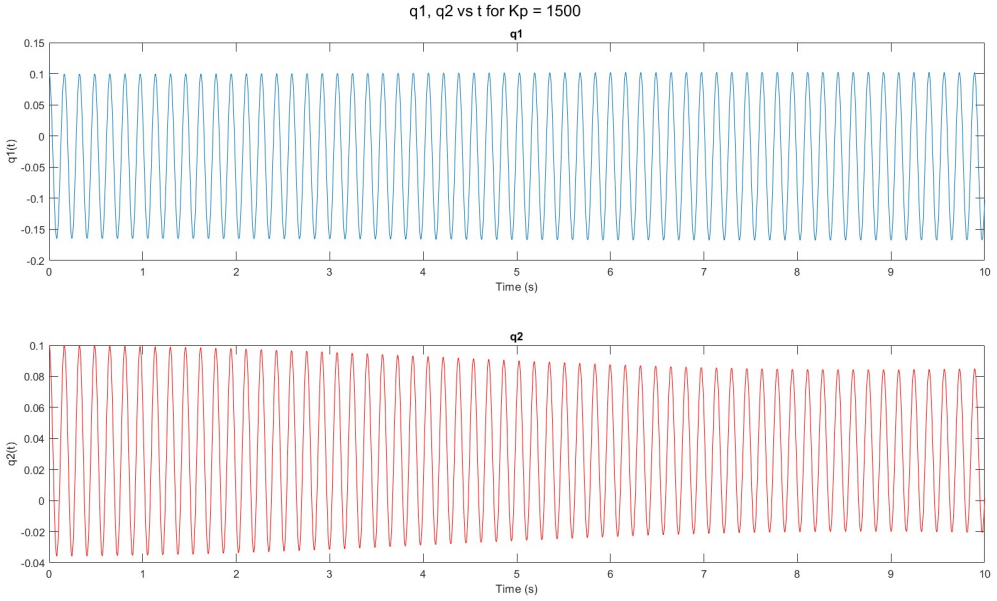
The control forces f1 and f2 for each joint are calculated using PID control with proportional, derivative, and integral terms. These control forces aim to minimize the joint angle errors.

The function returns a vector dxdt that represents the time derivatives of the states(used by ode45 to integrate the dynamics over time).

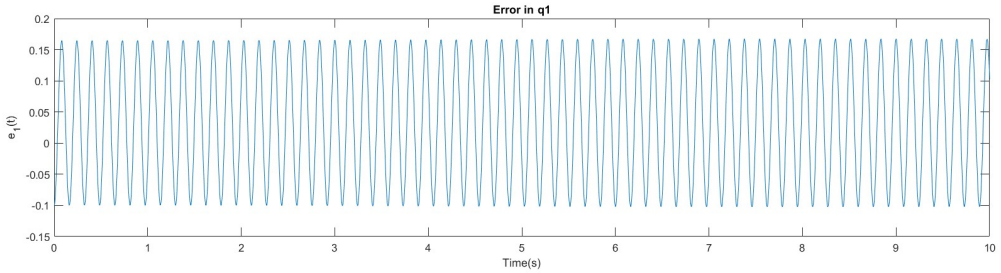
These derivatives are essential in describing the system's dynamic behavior.

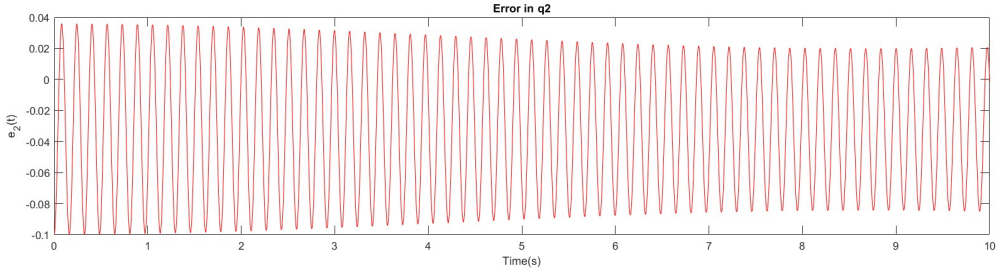
To sum up, this code sets up and simulates the control of a two-link planar manipulator using PID control. It computes the dynamics of the manipulator, calculates control forces, and evaluates the performance of the controller by plotting joint angles and errors over time.

P controller-

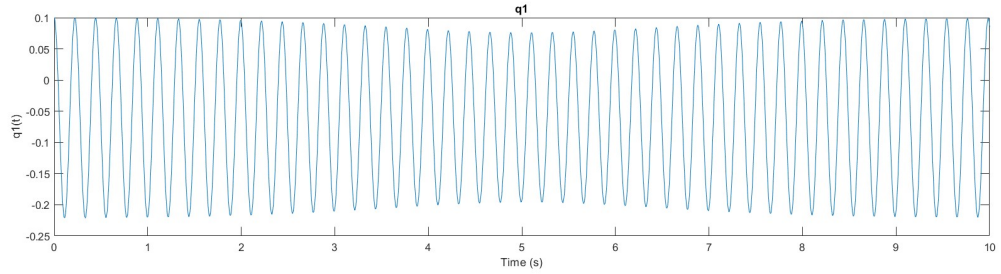


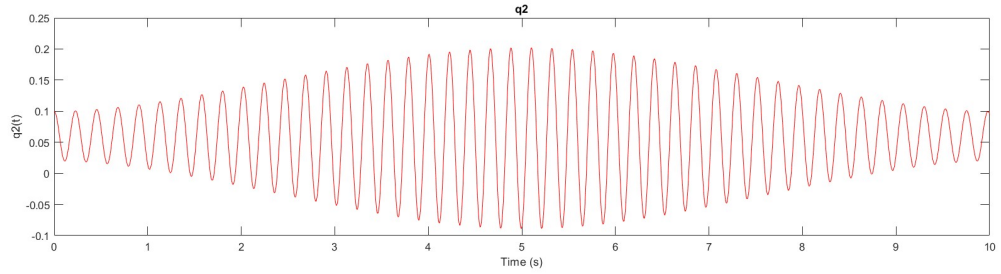
Error plots for Kp = 1500



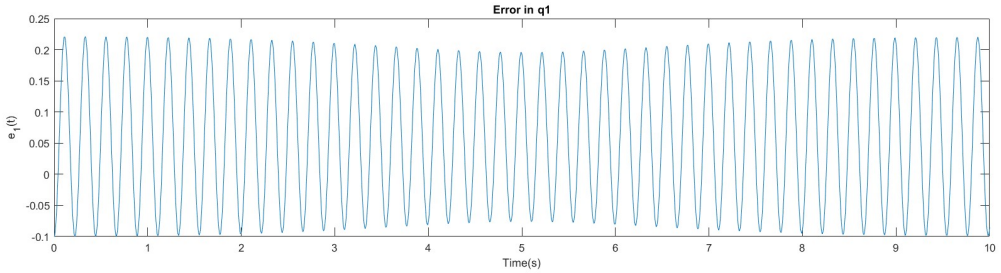


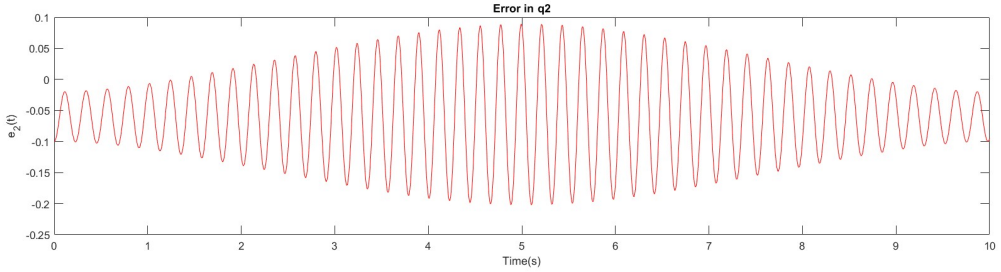
q1, q2 vs t for Kp = 800



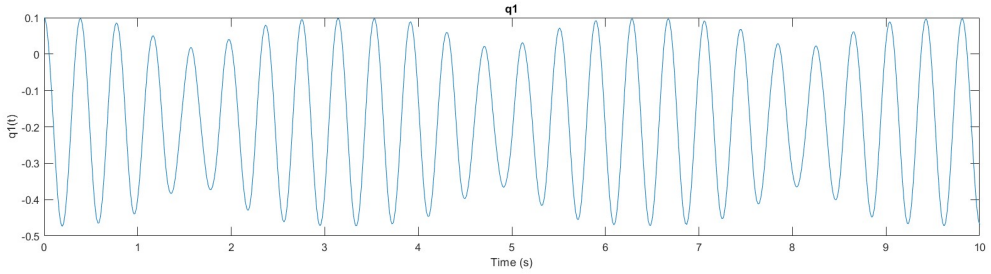


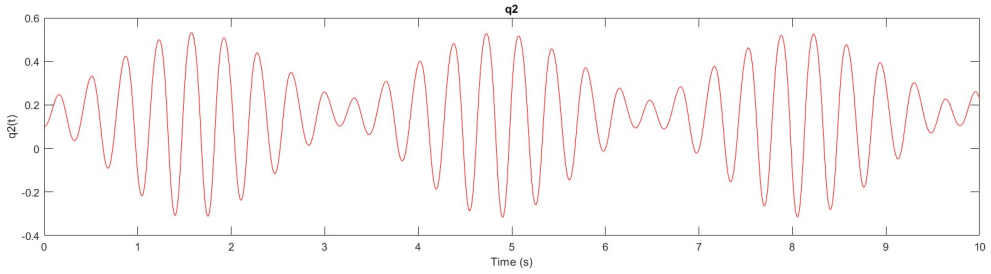
Error plots for Kp = 800



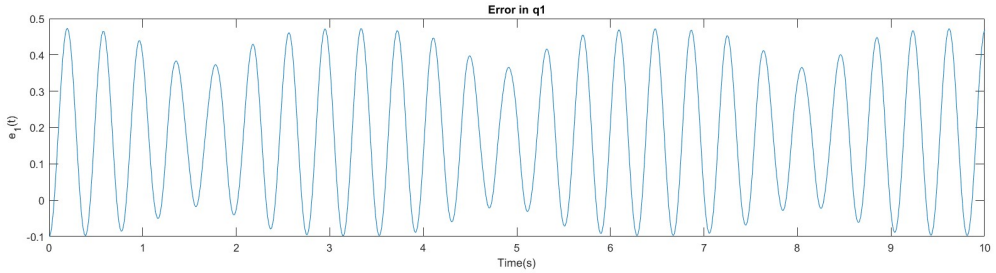


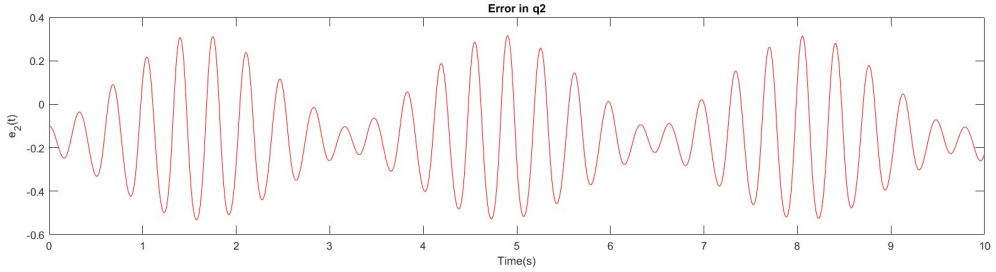
q1, q2 vs t for Kp = 250



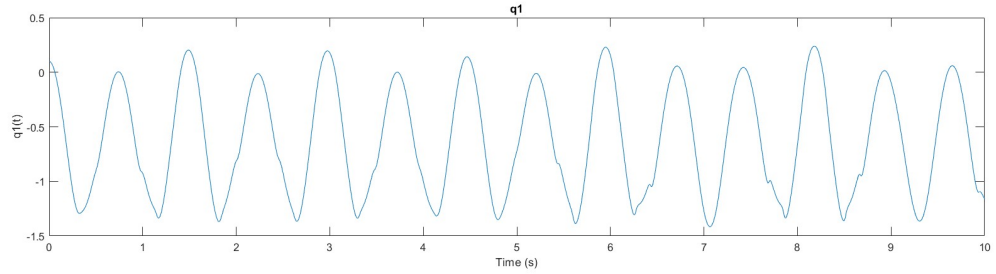


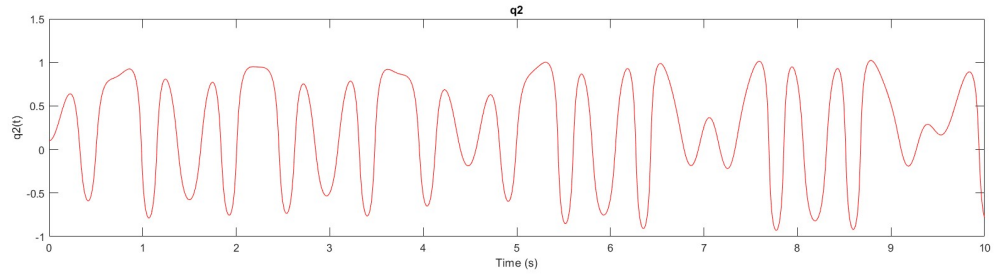
Error plots for Kp = 250

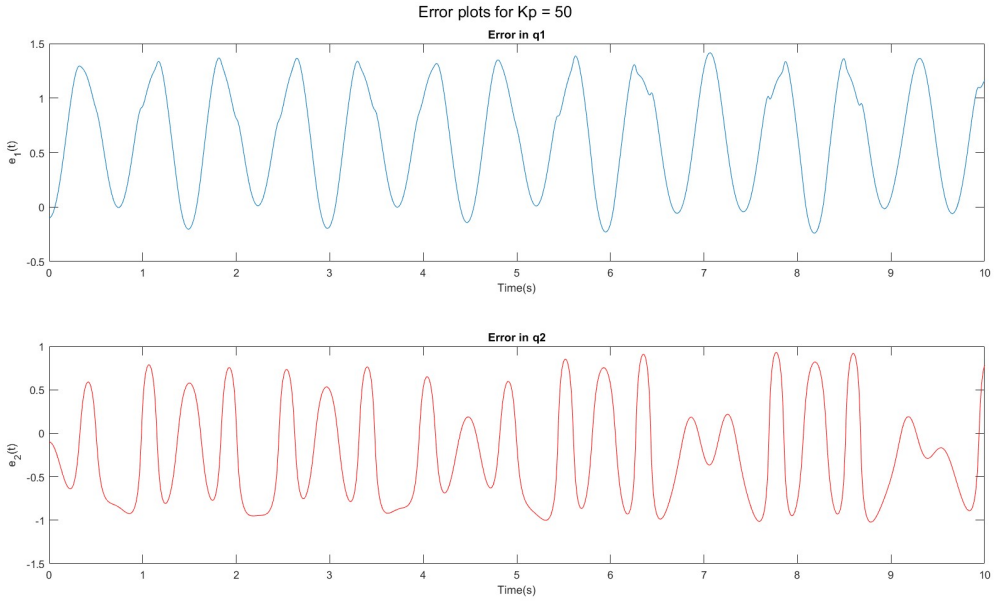




q1, q2 vs t for Kp = 50







Set both Ki and Kd to zero. Kp is initially set to a small value and then gradually increased.

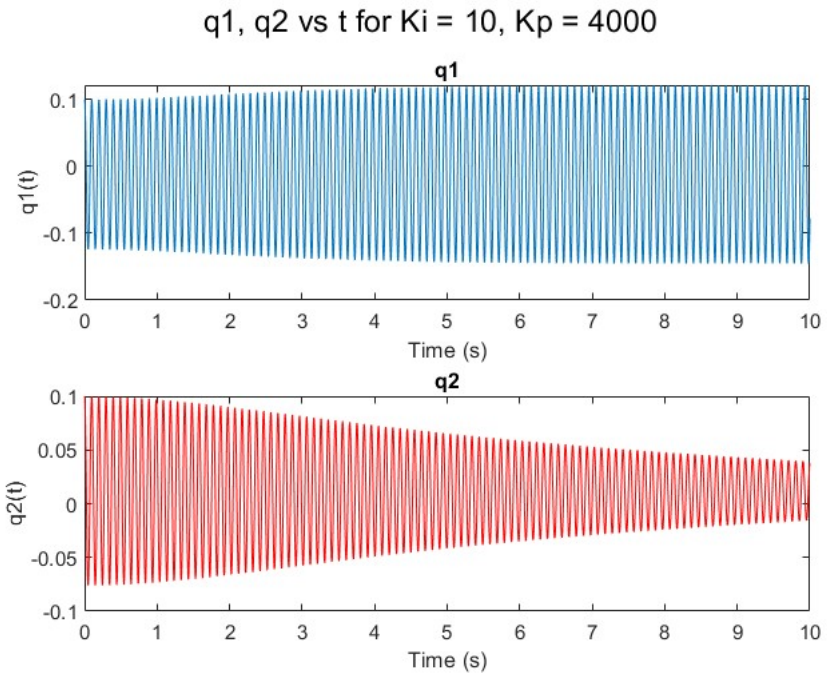
Increasing Kp makes the control action more aggressive.

If Kp is highly increased, the system becomes unstable and exhibits excessive oscillations.

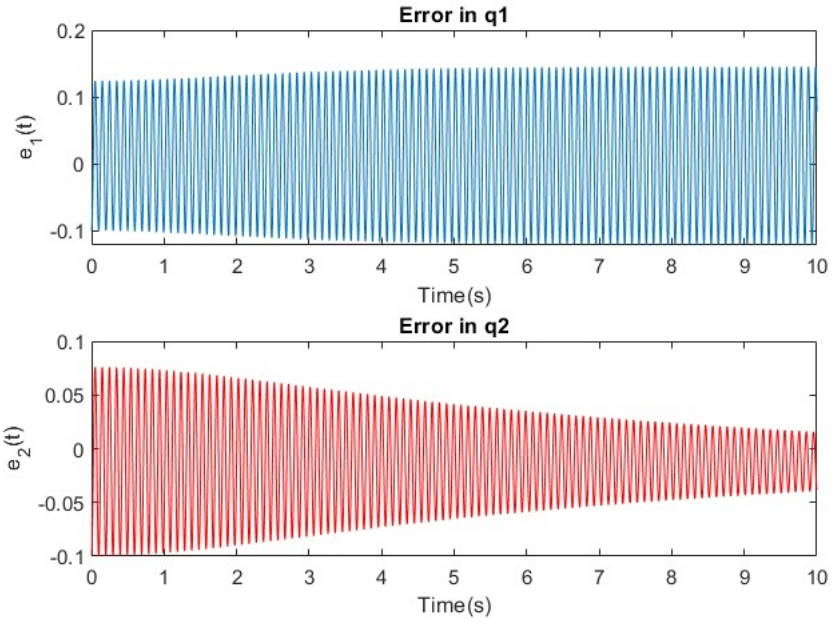
Reducing Kp adds more damping to the system response. The system response becomes slower as Kp is reduced.

We have to attempt to balance between this overshoot and settling time considering a compromise value for Kp.

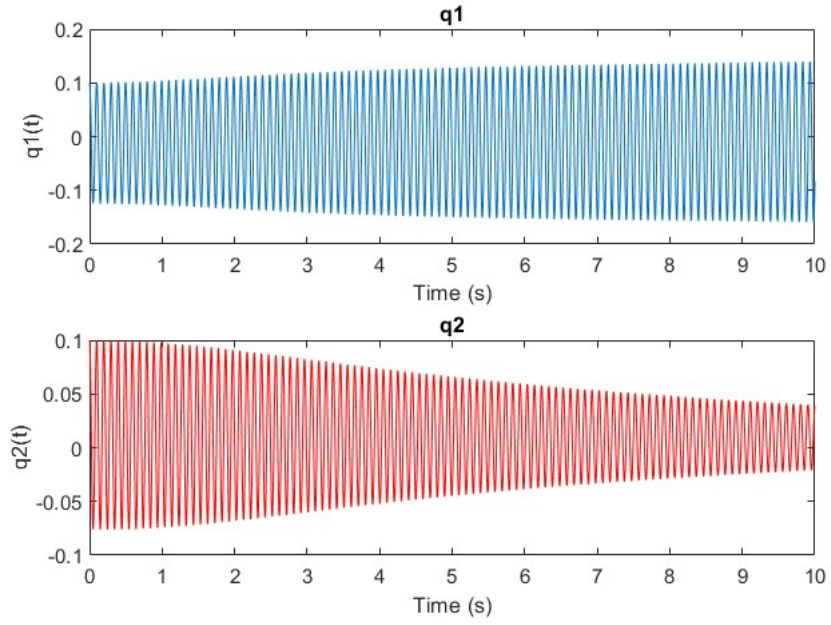
PI controller



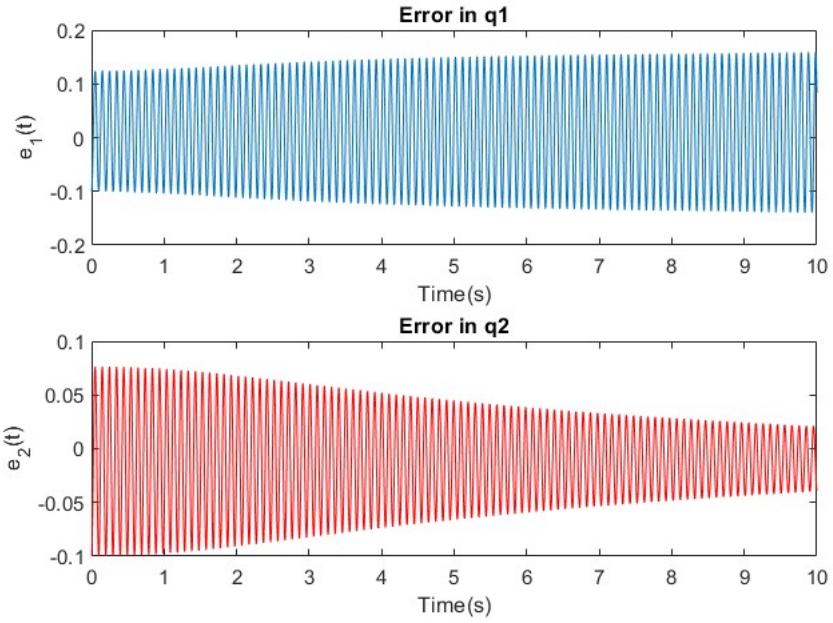
Error plots for Ki = 10, Kp = 4000



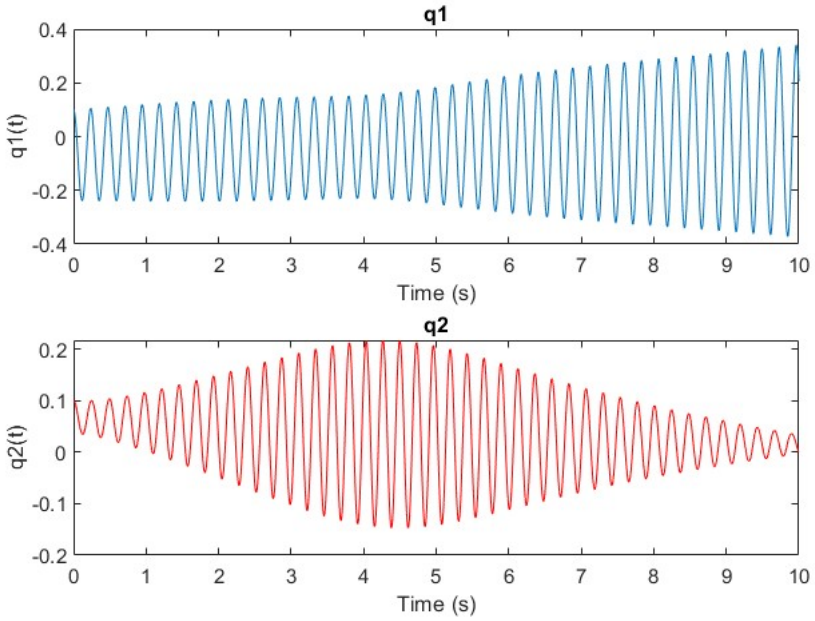
q1, q2 vs t for Ki = 100, Kp = 4000

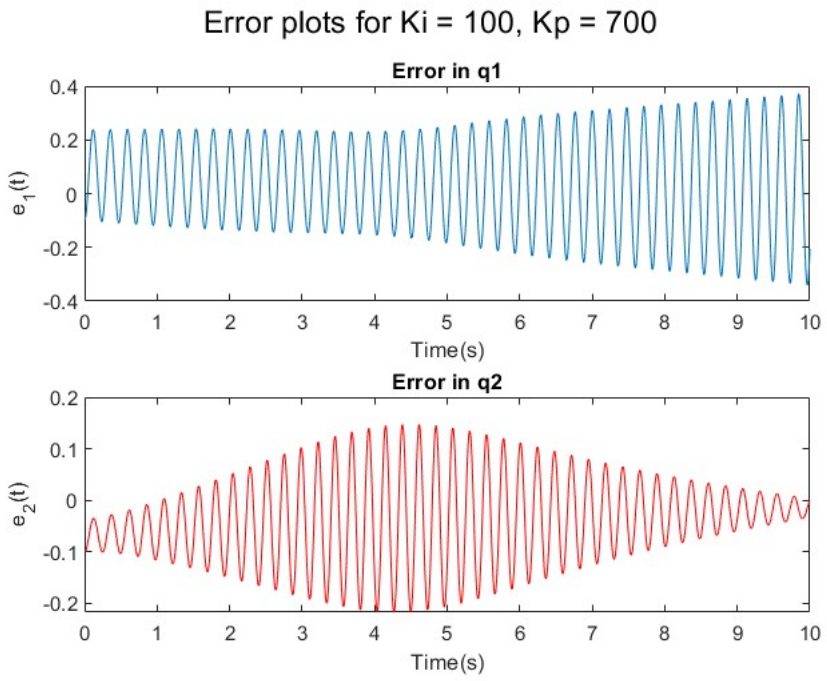


Error plots for Ki = 100, Kp = 4000



q1, q2 vs t for Ki = 100, Kp = 700



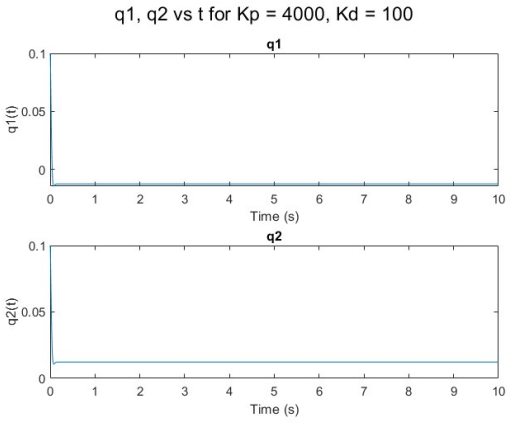


Integral control Ki is responsible for eliminating steady-state error. Increasing Ki leads to faster error correction which reduces steady-state deviations.Addition of a I-control helps eliminate the offset.But it can contribute to overshoot if set too high.The disadvantage is that it can destabilize the controller . I-only controllers are much slower in their response time. Small incremental adjustments to Ki should be made to ensure that it does not lead to excessive overshoot.

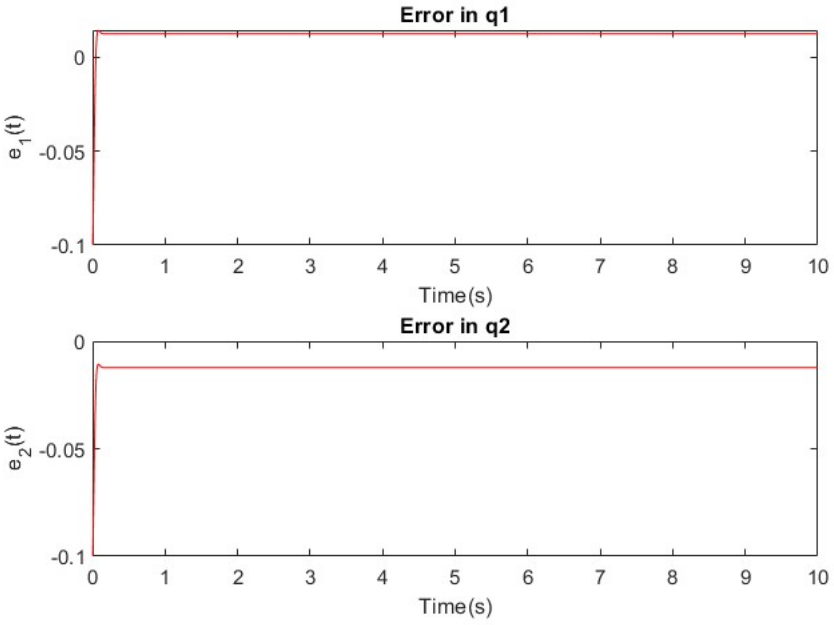
We initially give Kp=4k and change the value of ki for the first 2 plots to see the effct ki has on q1 and q2. On increasing Ki, the steady state error starts reducing and the system starts tracking the setpoint more accurately. An excessively high value of Ki can lead to excessive control action, which may result in instability or oscillations. As Ki was increased, the initial value of error was less but the error was diverging in nature.

A PI controller struggles to handle such nonlinearity effectively, leading to oscillations and steady-state errors.So it is generally not preferred.

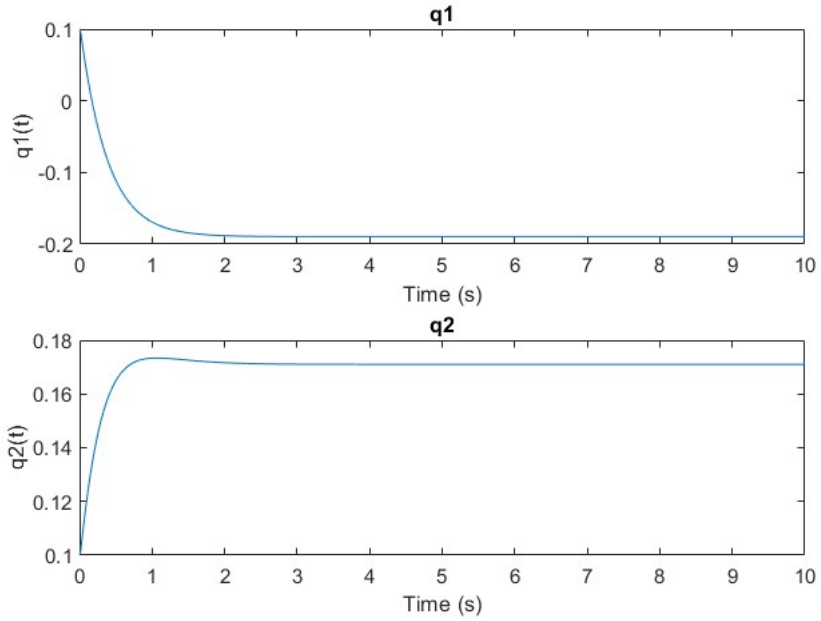
PD controller



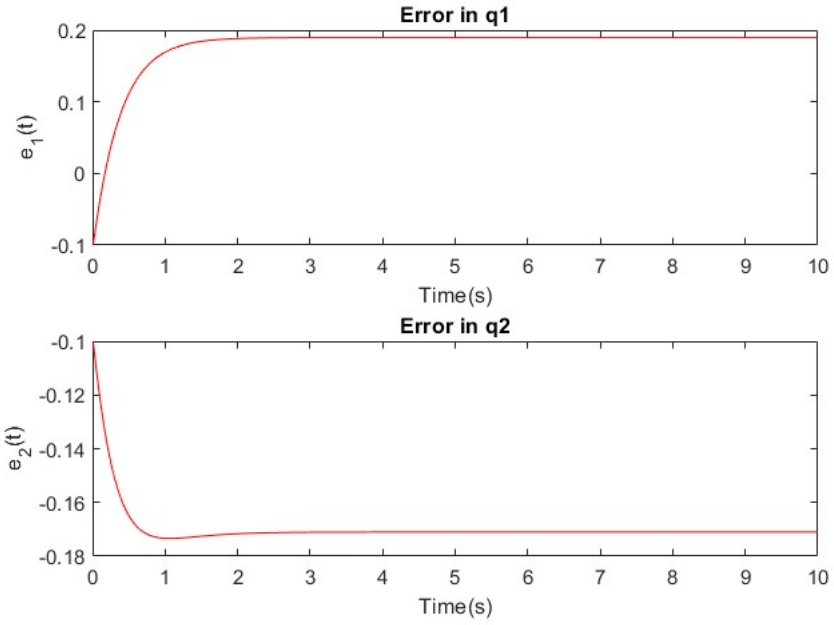
Error plots for Kp = 4000, Kd = 100



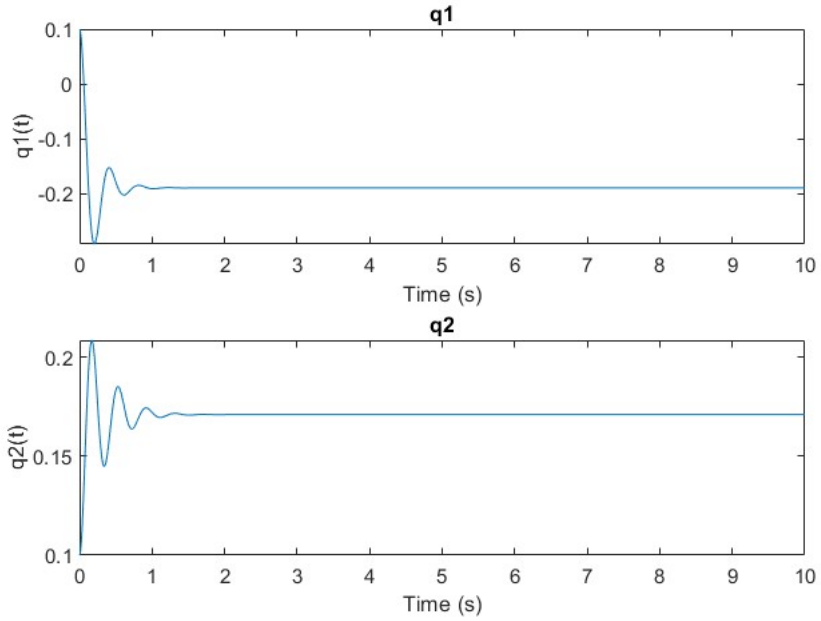
q1, q2 vs t for Kp = 250, Kd = 100

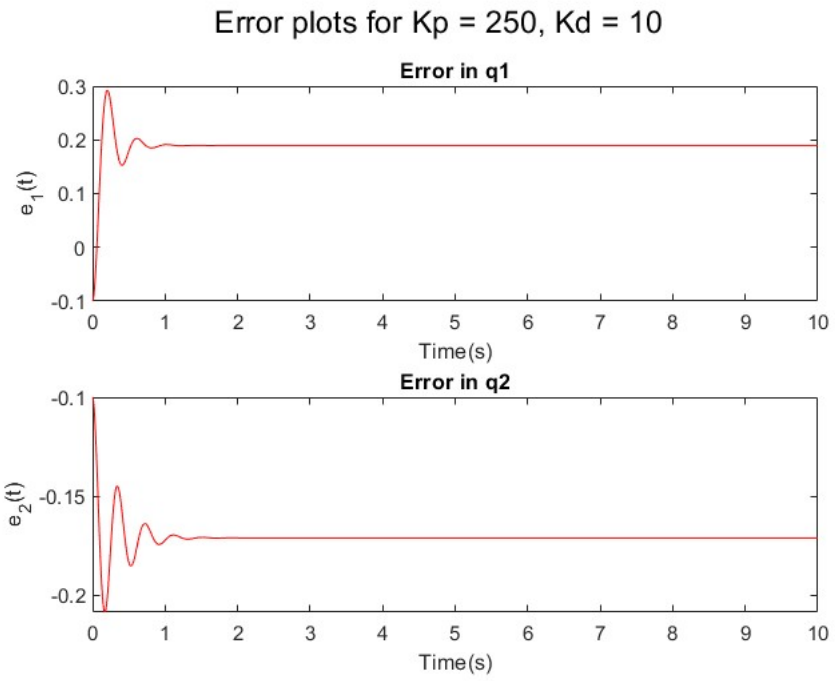


Error plots for Kp = 250, Kd = 100



q1, q2 vs t for Kp = 250, Kd = 10

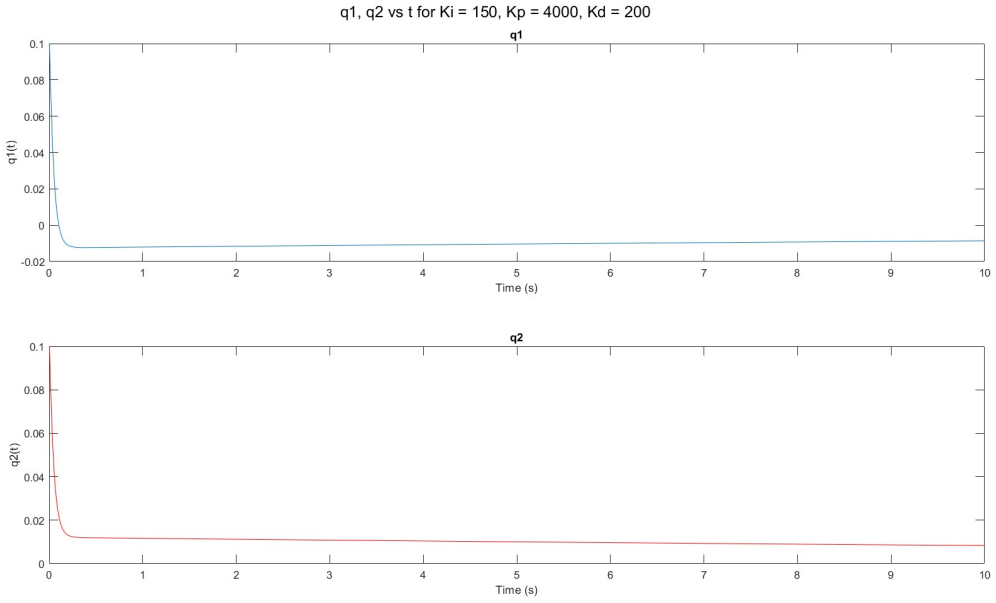




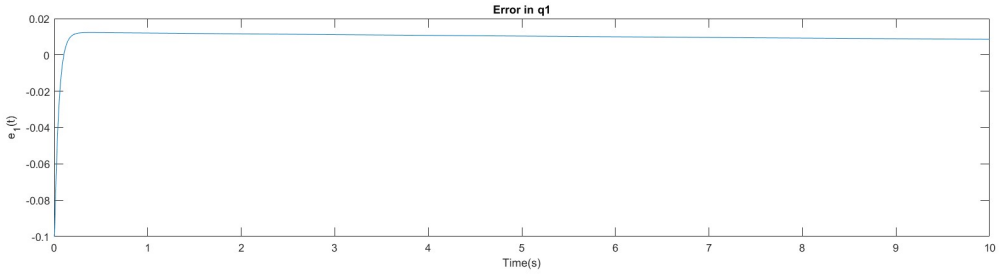
Derivative control Kd helps dampen oscillations by predicting how fast the error is changing. Increasing Kd stabilizes the system and reduces overshoot. Kd reduces settling time by reducing oscillations. The system reachs the setpoint more quickly without excessive oscillatory behaviour.

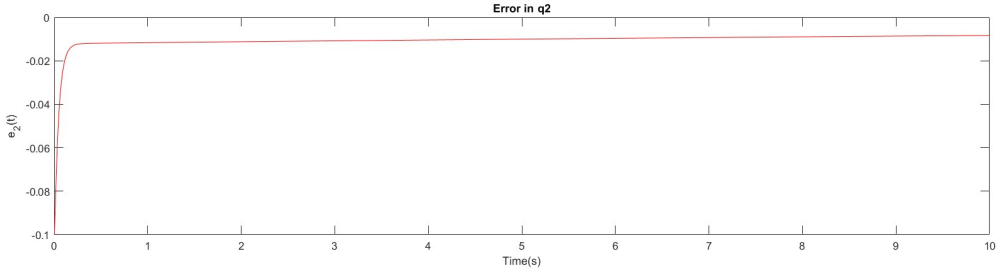
The last 2 graphs show how Kd affects the settling time and oscillations. As Kd is increased, we see a reduction in oscillations and overshoot. Kd dampens the system which can improve the transient response and reduce oscillatory behaviour. Excessively high Kd value can lead to instability and noise amplification in the control system.The optimal Kd value results in a controlled and well-damped response without excessive oscillations.

PID controller

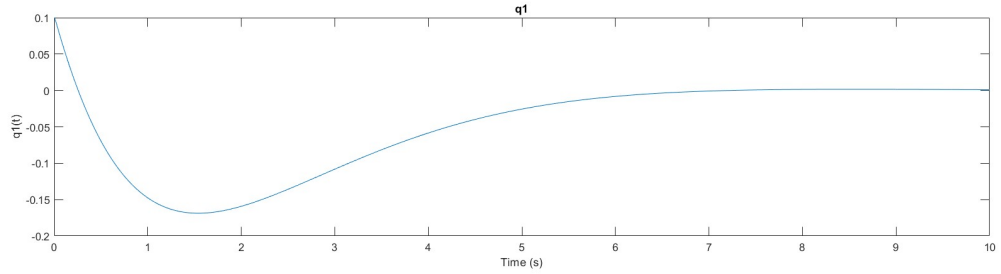


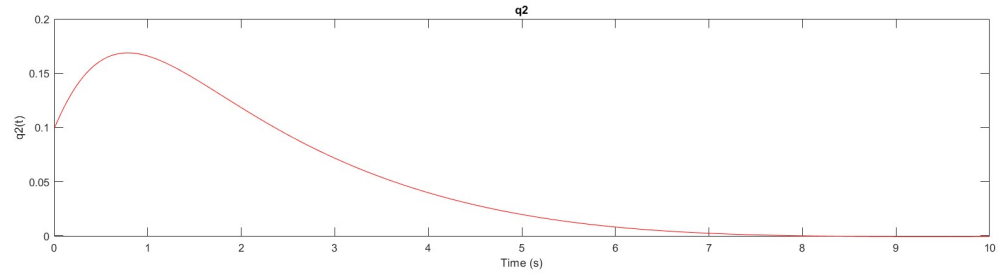
Error plots for Ki = 150, Kp = 4000, Kd = 200



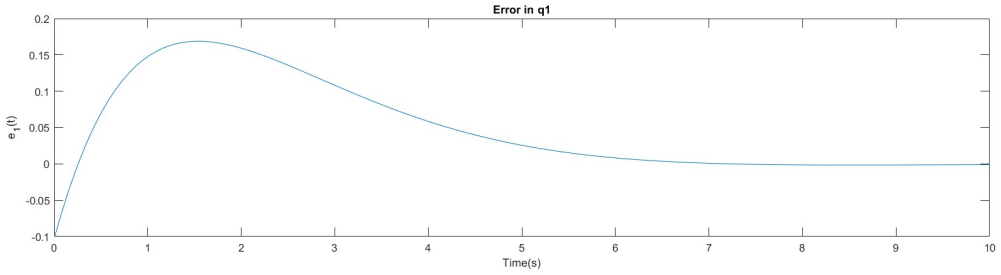


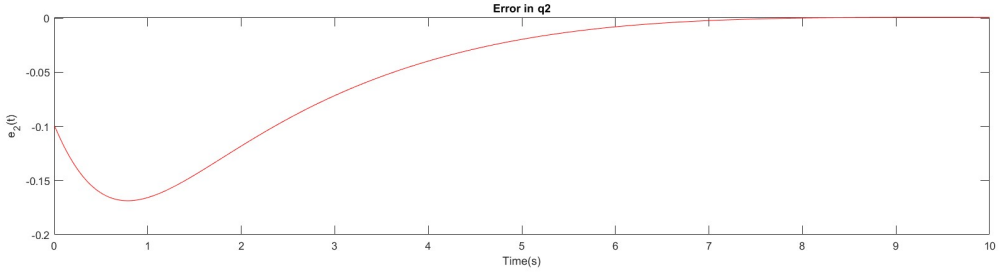
q1, q2 vs t for Ki = 100, Kp = 200, Kd = 150



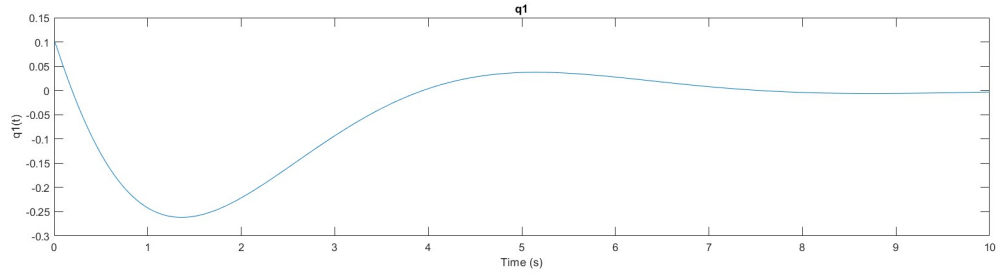


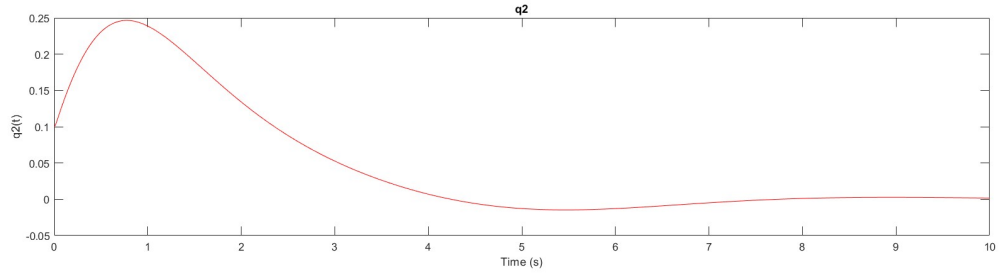
Error plots for Ki = 100, Kp = 200, Kd = 150

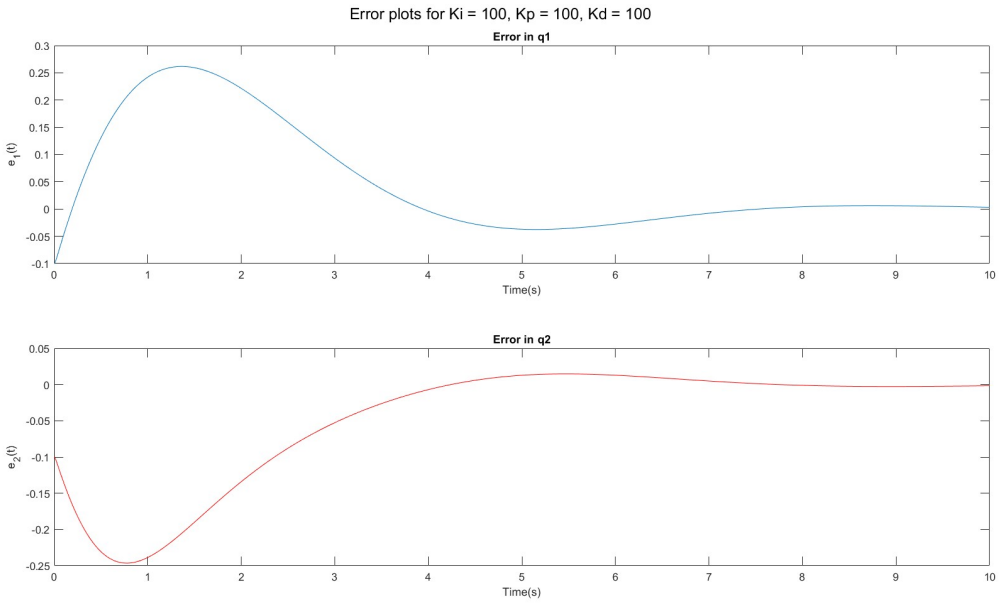




q1, q2 vs t for Ki = 100, Kp = 100, Kd = 100





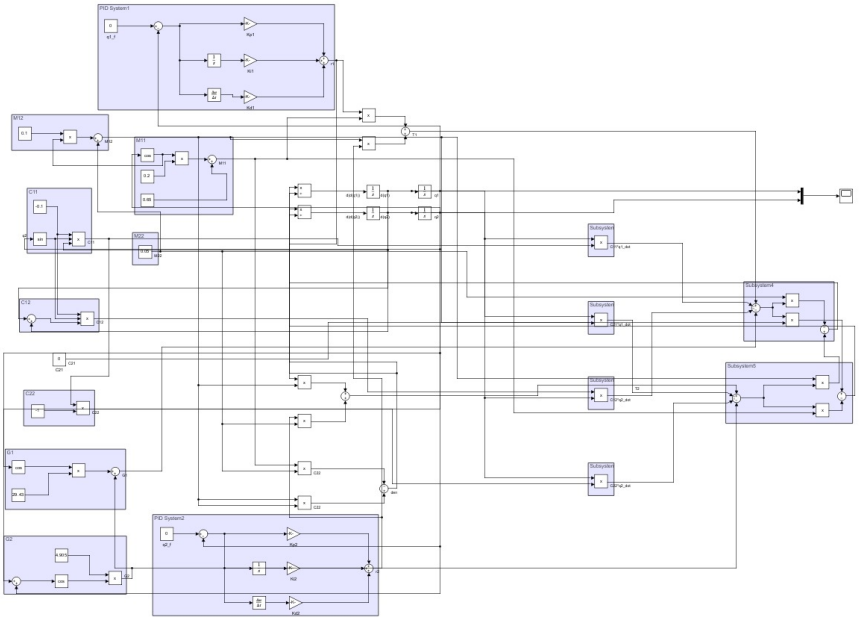


We can clearly see the difference between PID and other controllers from these graphs .There are no excessive oscillations and the settling time is also good.

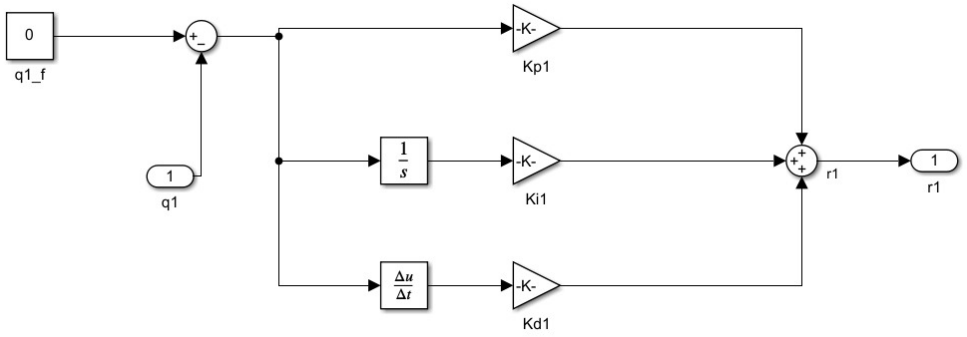
We need to assign a suitable value for Kp to balance between overshoot and settling time and setting both Kd and Ki to zero. Then we can gradually increase Kd to reduce overshoot and improve damping. We continue to adjust Kd until we find a balance between reducing overshoot and maintaining stability. We obtain a highly stable system response with quite less overshoot and settling time but some steady state error. To
reduce this steady state error, we gradually increase Ki until we obtain a system response with negligible steady state error. Thus, we obtain a highly stable system response with negligible error which attains the final state quickly without deviating much from the desired path.Kp value 200, Kd value 150 and Ki value 100 give out the best results for the PID controller.

SIMULINK:

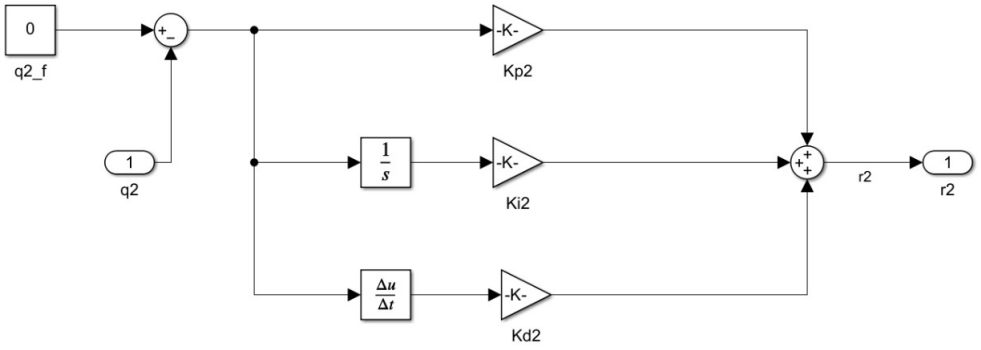
The Block Diagram Of The System Designed:



PID Controller -1 Block Diagram

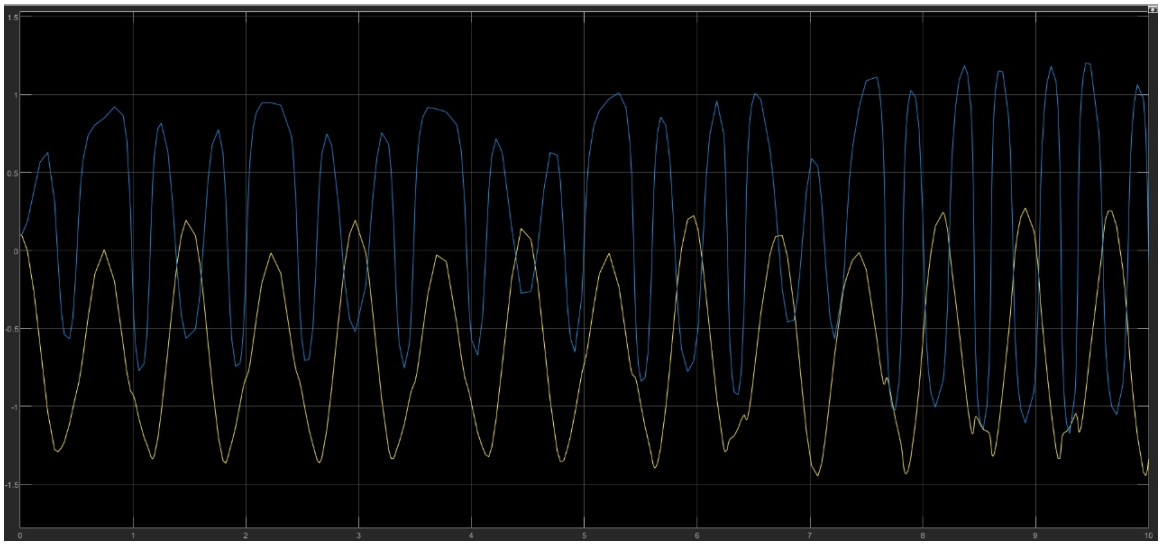


PID Controller - 2 Block Diagram

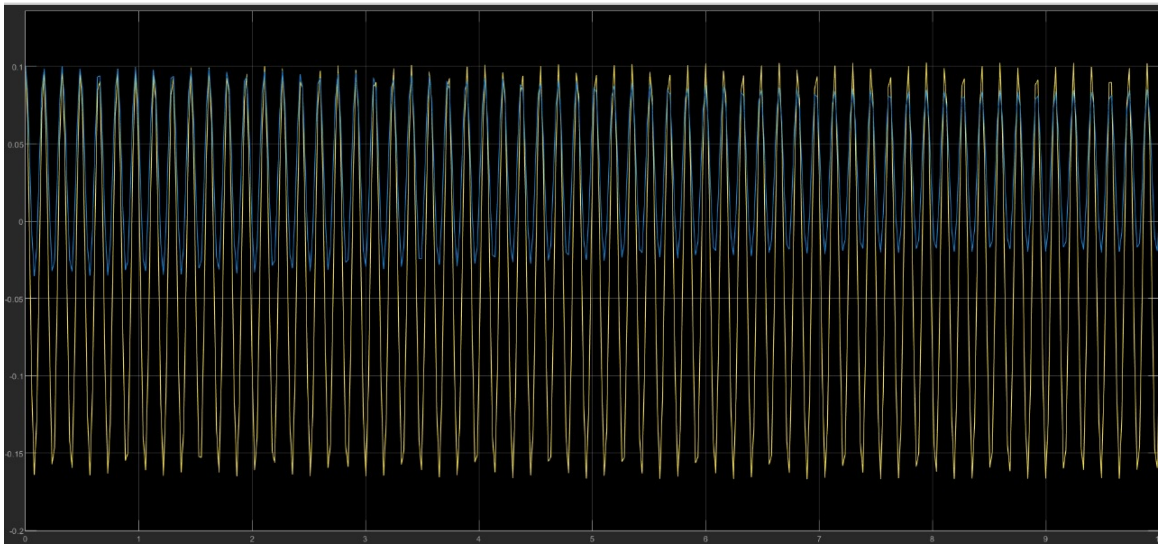


P Controller

For $K_p = 50, K_d = 0, K_i = 0$

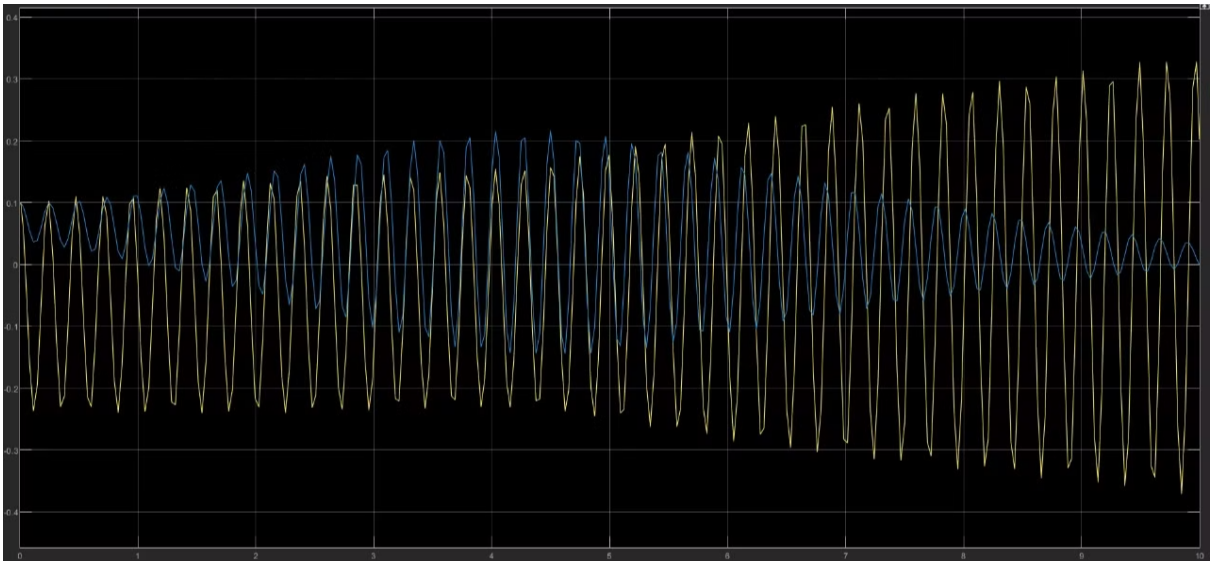


For $K_p = 1500, K_d = 0, K_i = 0$

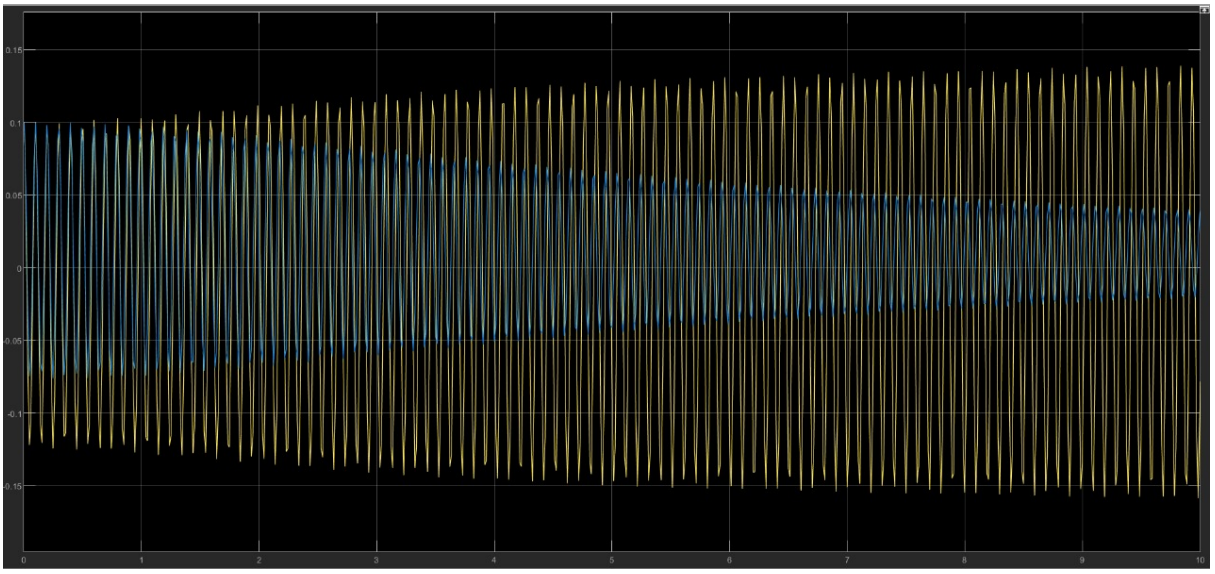


PI Controller

For $K_p = 700, K_d = 0, K_i = 100$

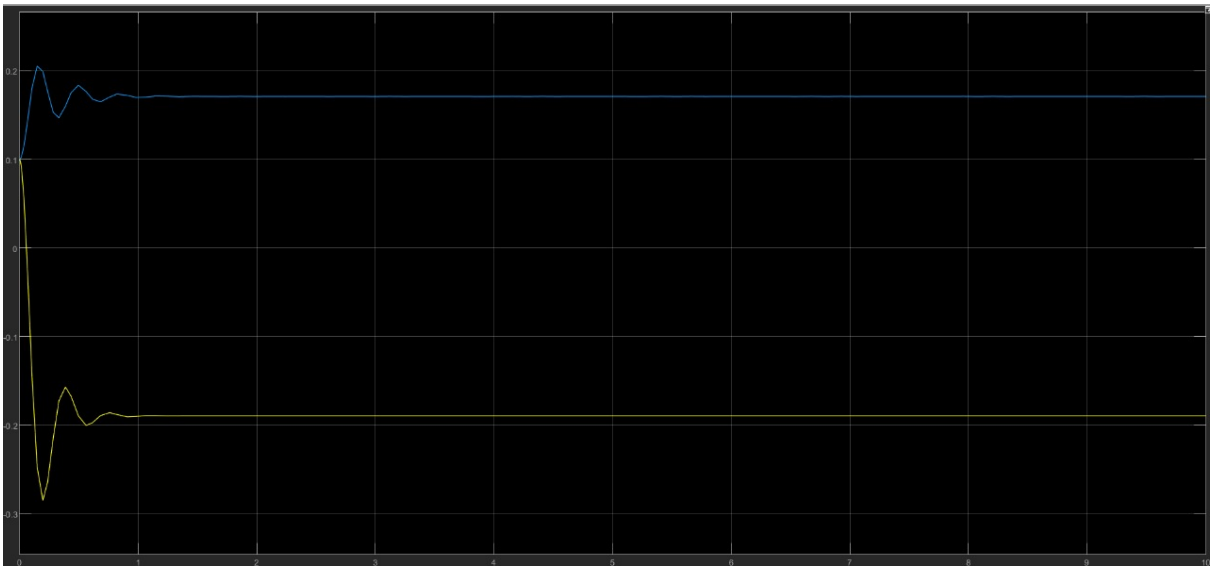


For $K_p = 4000, K_d = 0, K_i = 100$

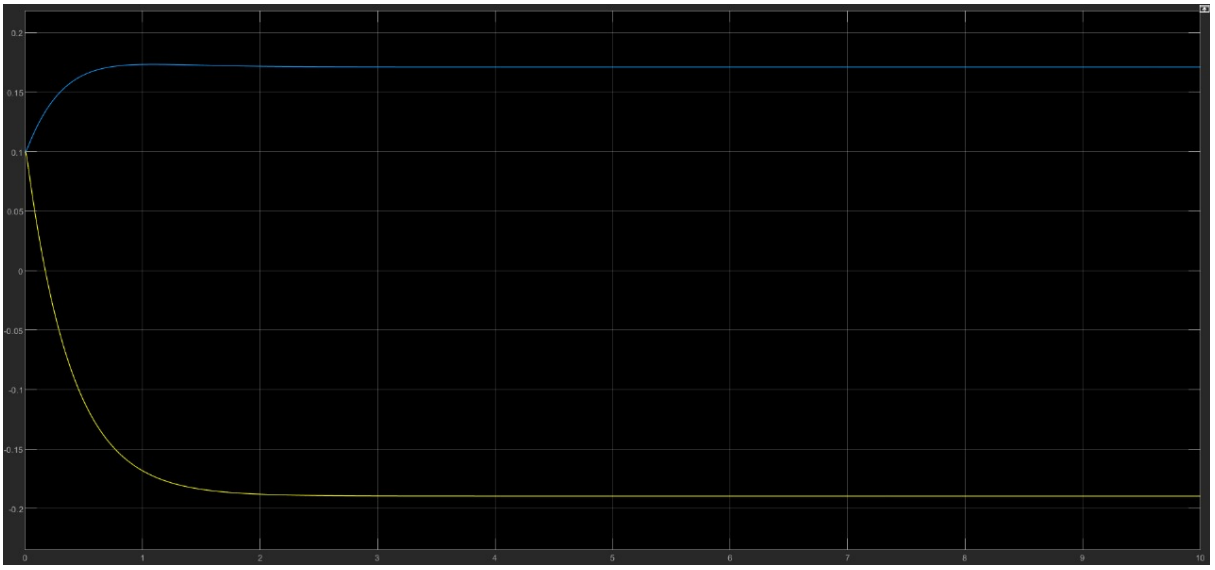


PD Controller

For $K_p = 250, K_d = 10, K_i = 0$

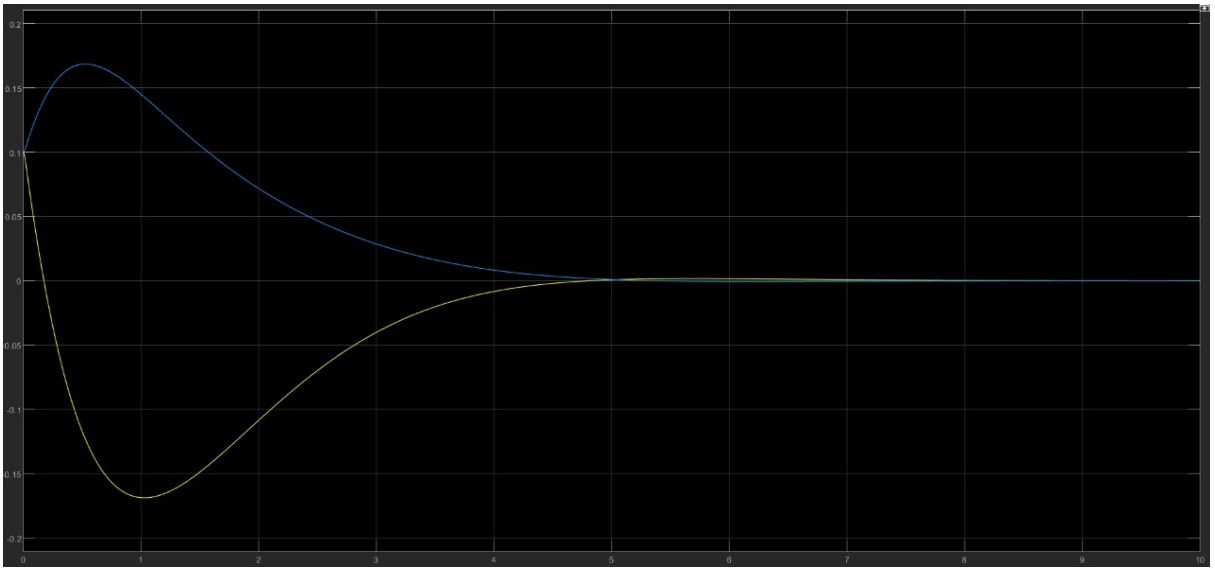


For $K_p = 250, K_d = 100, K_i = 0$

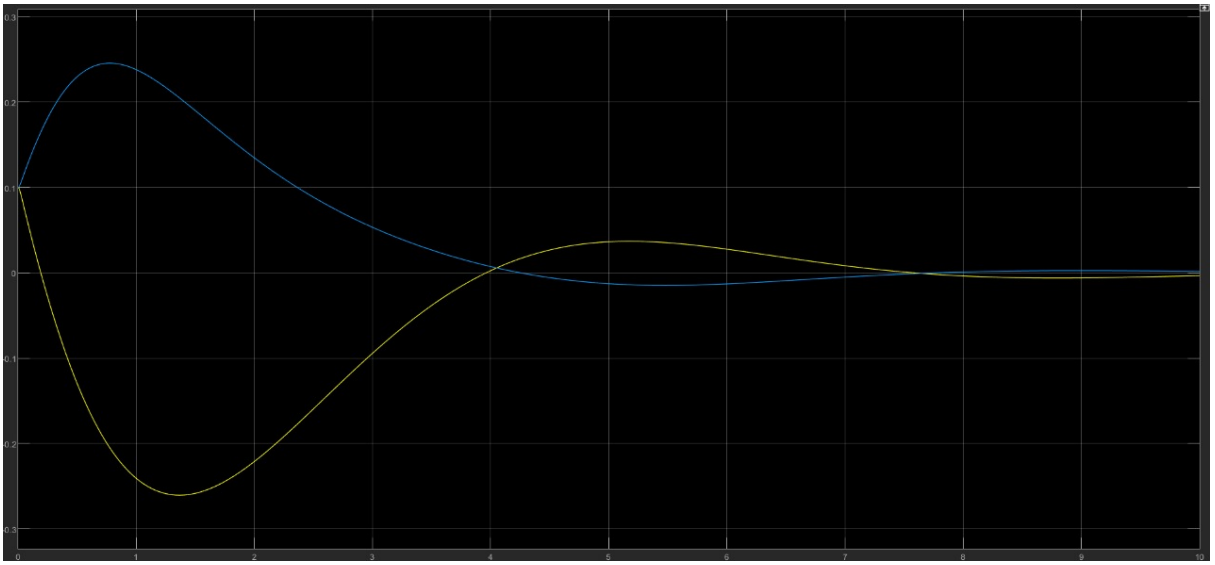


PID Controller

For $K_p = 200, K_d = 100, K_i = 150$



For $K_p = 100, K_d = 100, K_i = 100$



CONCLUSION:

The performance of a two-link robotic manipulator was evaluated using three control strategies: Proportional (P), Proportional-Derivative (PD), and Proportional-Integral-Derivative (PID) controllers. The PID controller was found to offer the best overall performance, providing accurate trajectory tracking with minimal overshoot and minimum steady-state error.

While the PD controller exhibited larger overshoot and steady-state error, the PI controller effectively eliminated the steady-state error but at the cost of more oscillations. The dynamics of the robot were modeled using the Euler-Lagrange method, and simulations demonstrated that PID tuning resulted in smoother, more precise control.

The iterative tuning of the controller parameters (Kp, Ki, Kd) was essential for achieving optimal control performance, with the PID controller showing its adaptability by functioning as a PI or PD controller as needed. Overall, the results suggest that PID control is a robust and effective solution for controlling the motion of robotic systems, providing enhanced stability and accuracy compared to simpler control methods.

In conclusion:

- K_p addresses the current error, providing immediate correction.
- K_i addresses cumulative past errors, ensuring that the system reaches the desired setpoint without a steady-state offset.
- K_d addresses future errors by smoothing the response, preventing overshoot or oscillations.

A well-tuned PID controller balances these three terms to optimize system performance for stability, accuracy, and response time.