Systems Thinking Project

2-Link Manipulator

Team: Blank...

Diddigi Srivatsa	2025122009
Devang Bordoloi	2025122003
Mohammed Adeeb	2025122005
Mididoddi Poojith	2025122010
Yogesh Krishna	2024102035
Prakash Karedla	2025122006

Contents

1 Dynamics

1.1 Analysis of the dynamics using Lagrangian Mechanics
- 1.1.1 Finding Total Kinetic Energy of the system
- 1.1.2 Finding Potential Energy(U) of the system
- 1.1.3 Expression of Lagrangian of the system
- 1.1.4 Extracting equations of motion from the Lagrangian
- 1.1.5 Obtaining given second-order equations of motion

1.2 State-space modelling
- 1.2.1 Obtaining State-space equation

2 Controller Design for the 2-Link Manipulator

2.1 Prerequisites: Dynamic Decoupling of the Manipulator
2.2 Proportional-Derivative (PD) Control
2.3 Proportional-Integral (PI) Control
2.4 Proportional-Integral-Derivative(PID) Control

3 The TwoLink_PID_Dynamics Code

3.1 Proportional Derivative Controller
- 3.1.1 Code
- 3.1.2 Plots

3.2 Proportional Integral Controller
- 3.2.1 Code
- 3.2.2 Plots

3.3 Proportional Integral Derivative Controller
- 3.3.1 Code
- 3.3.2 Plots

4 SIMULINK

4.1 Block Diagram
4.2 Proportional Integral Derivative Controller
4.3 Proportional Integral Controller
4.4 PID Controller

5 Conclusion

5.1 Proportional Derivative Controller
- 5.1.1 Effect of Derivative Gain (Kd)
- 5.1.2 Conclusion

5.2 Proportional Integral Controller
- 5.2.1 Effect of Proportional Gain (Kp)
- 5.2.2 Effect of Integral Gain (Ki)
- 5.2.3 Conclusion

5.3 Proportional Integral Derivative Controller
- 5.3.1 Effect of Proportional Gain (Kp)
- 5.3.2 Effect of Derivative Gain (Kd)
- 5.3.3 Effect of Integral Gain (Ki)
- 5.3.4 Conclusion

Abstract

In this study, we design and evaluate a Proportional-Integral-Derivative (PID) controller for a classical two-link robotic manipulator. The dynamic equations of motion for the manipulator are first derived to establish the system model. The main objective is to achieve precise position control of the manipulator using the computed torque control technique. The control strategies are implemented and tested through MATLAB simulations. Multiple simulation experiments are conducted to assess the system's performance under Proportional-Derivative (PD), Proportional-Integral (PI), and Proportional-Integral-Derivative (PID) control schemes with varying controller gains (Kp, Ki, Kd). The effectiveness and robustness of the proposed controller are validated using MATLAB Simulink, demonstrating reliable tracking and improved control performance.

1. DYNAMICS:

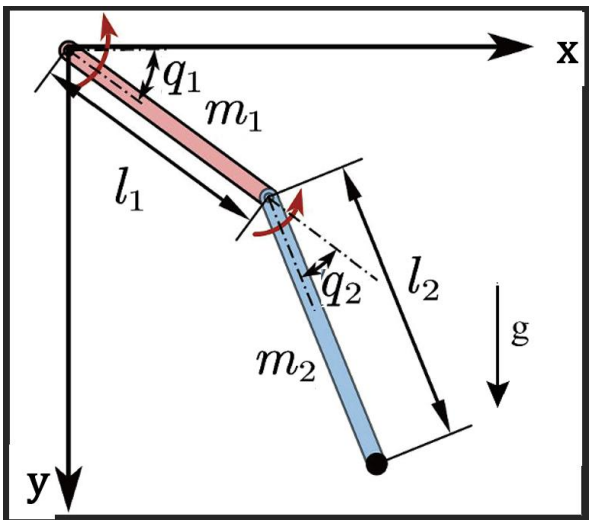


Figure 1: Diagram of the two-link manipulator with real rods (with mass)

The two-link manipulator system, as can be seen in the diagram is a system of two ideal(massless) rods connected adjacent to each other and are driven by two external torques acting on the two joints in the system. Two point-masses are considered at the ends of the rods m_1 and m_2, rods being of lengths l_1 and l_2 as can be seen from the figure 2. Angle q_1 denotes the angle that the rod of length l_1 makes with the positive x axis and q_2 denotes the angle that the rod of length l_2 makes with the axis of rod of length l_1. The two-link manipulator system describes its motion in a way that angles q_1 and q_2 denote the 2 degrees of freedom of motion of the two rotating rods. Torques τ_1 and τ_2 are the external torques applied to the system of two-link manipulator to give rise to the motion.

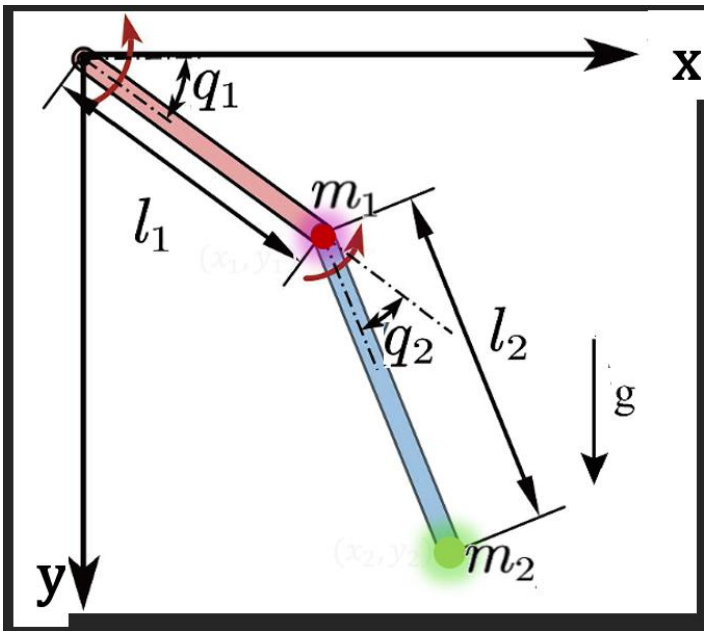


Figure 2: Diagram considering ideal massless rods(used in our analysis)

1.1: Analysis of the dynamics using Lagrangian Mechanics:

We have analyzed the motion of given two-link manipulator system using Lagrangian mechanics, this is done due to the following reasons:

- Lagrangian mechanics allows us to analyze dynamics of multi-particle systems, motions of particles involving constraints and it allows us to analyze dynamics of systems in cases where fictious forces (Coriolis force and/or Centrifugal force) make the formulation of force dynamics equations difficult.
- The Lagrangian method relies on energy-analysis (finding kinetic and potential energies of the system; Lagrangian $(L) = T - U$) which simplifies analysis greatly.
- Lagrangian analysis allows us to use generalized coordinates for analysis in cases where calculations in cartesian coordinates becomes impractical.

We analyze the motion of the two-link manipulator using Lagrangian Mechanics in the following way:

Using the fact that the Lagrangian is defined as:

$$L = T - U$$

Where, T denotes the Kinetic Energy of the system,

and U is the total Potential Energy of the system

1.1.1: Finding Total Kinetic Energy of the system:

For finding the Lagrangian, firstly, we find the Kinetic Energy (T) of the system in the following way:

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

We express the coordinates of masses $m_1$and m_2 that are (x_1, x_2) and (x_2, y_2) respectively, in terms of l_1, q_1, l_2, q_2 where l_1 and l_2 are constants:

$$x_1 = l_1 \cos q_1$$

$$y_1 = -l_1 sinq_1$$

$$x_2 = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$y_2 = -l_1 sinq_1 - l_2 \sin(q_1 + q_2)$$

Kinetic Energy(T) of the system is found as the following expression:

$$T = \frac{1}{2}m_1((-l_1 sinq_1\dot{q_1})^2 + (-l_1 cosq_1\dot{q_1})^2)$$
$$+\frac{1}{2}m_2((-l_1 sinq_1\dot{q_1} + -l_2 sin(q_1 + q_2)(\dot{q_1} + \dot{q_2}))^2 + (-l_1 cosq_1\dot{q_1} + -l_2 cos(q_1 + q_2)(\dot{q_1} + \dot{q_2}))^2)$$

$$T = \frac{(m_1 + m_2)}{2}(l_1\dot{q_1})^2 + \frac{m_2}{2}l_2^2(\dot{q_1} + \dot{q_2})^2 + m_2 l_1 l_2 \dot{q_1}(\dot{q_1} + \dot{q_2})cosq_2$$

1.1.2: Finding Potential Energy(U) of the system:

Next, we obtain the Gravitational Potential Energy (U) of the system in the following way:

$$U = m_1 g y_1 + m_2 g y_2$$

The expression that we obtain for the gravitational potential energy is the following:

$$U = -m_1 g l_1 sinq_1 - m_2 g(l_1 sinq_1 + l_2 \sin(q_1 + q_2))$$

1.1.3: Expression of Lagrangian of the system:

The expression of the Lagrangian for the system's motion is:

$$L = T - U$$

$$L = \frac{(m_1 + m_2)}{2}(l_1\dot{q_1})^2 + \frac{m_2}{2}l_2^2(\dot{q_1} + \dot{q_2})^2 + m_2 l_1 l_2 \dot{q_1}(\dot{q_1} + \dot{q_2})cosq_2 + m_1 g l_1 sinq_1 + m_2 g(l_1 sinq_1 + l_2 \sin(q_1 + q_2))$$

$$L = \frac{(m_1 + m_2)}{2}(l_1\dot{q_1})^2 + \frac{m_2}{2}l_2^2(\dot{q_1} + \dot{q_2})^2 + m_2 l_1 l_2 \dot{q_1}(\dot{q_1} + \dot{q_2})cosq_2 + (m_1 + m_2) g l_1 sinq_1 + m_2 g l_2 \sin(q_1 + q_2)$$

1.1.4: Extracting equations of motion from the Lagrangian:

We obtained the Lagrangian which was a function:

$$L(l_1, q_1, \dot{q_1}, l_2, q_2, \dot{q_2})$$

But since l_1 and l_2 are constants, the lagrangian is essentially a function of $q_1, \dot{q_1}, q_2$ and $\dot{q_2}$ only i.e.:

$$L(q_1, \dot{q_1}, q_2, \dot{q_2}) = \frac{(m_1 + m_2)}{2}(l_1\dot{q_1})^2 + \frac{m_2}{2}l_2^2(\dot{q_1} + \dot{q_2})^2 + m_2 l_1 l_2 \dot{q_1}(\dot{q_1} + \dot{q_2})cosq_2 + (m_1 + m_2) g l_1 sinq_1 + m_2 g l_2 \sin(q_1 + q_2)$$

For a joint i, where external Torque τ_i is acting, the Euler-Lagrange equation of motion will be:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q_i}}\right) - \frac{\partial L}{\partial q_i} = \tau_i$$

Applying the above equation on the two joints 1 and 2, we have:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q_1}}\right) - \frac{\partial L}{\partial q_1} = \tau_1 \qquad \ldots (eq.1)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q_2}}\right) - \frac{\partial L}{\partial q_2} = \tau_2 \qquad \ldots (eq.2)$$

Evaluating eq.1:

$$\frac{d}{dt}((m_1 + m_2)l_1^2\dot{q_1} + m_2 l_2^2(\dot{q_1} + \dot{q_2}) + m_2 l_1 l_2 \dot{q_1} cosq_2 + m_2 l_1 l_2(\dot{q_1} + \dot{q_2})cosq_2) - (-(m_1 + m_2) g l_1 cosq_1 - m_2 g l_2 \cos(q_1 + q_2)) = \tau_1$$

$$\tau_1 = (m_1 + m_2)l_1^2\ddot{q_1} + m_2 l_2^2(\ddot{q_1} + \ddot{q_2}) + m_2 l_1 l_2 \ddot{q_1} cosq_2 - m_2 l_1 l_2 \dot{q_1}\dot{q_2} sinq_2 + m_2 l_1 l_2(\ddot{q_1} + \ddot{q_2})cosq_2 - m_2 l_1 l_2 \dot{q_2}(\dot{q_1} + \dot{q_2})sinq_2 + (m_1 + m_2) g l_1 cosq_1 + m_2 g l_2 \cos(q_1 + q_2) \qquad \text{(eqs.3)}$$

Now, evaluating eq.2:

$$\frac{d}{dt}(m_2 l_2^2(\dot{q_1} + \dot{q_2}) + m_2 l_1 l_2 \dot{q_1} cosq_2) - (-m_2 l_1 l_2 \dot{q_1}(\dot{q_1} + \dot{q_2})sinq_2 + m_2 g l_2 \cos(q_1 + q_2)) = \tau_2$$

$$\tau_2 = m_2 l_2^2(\ddot{q_1} + \ddot{q_2}) + m_2 l_1 l_2 \ddot{q_2} cosq_2 + m_2 l_1 l_2 \dot{q_1}^2 sinq_2 - m_2 g l_2 \cos(q_1 + q_2) \qquad \text{(eqs.4)}$$

What is given in the document we have used the below equation due to time constraints:

$$\tau_2 = m_2 l_2^2(\ddot{q_1} + \ddot{q_2}) + m_2 l_1 l_2 \ddot{q_1} cosq_2 + m_2 l_1 l_2 \dot{q_1}\dot{q_2} sinq_2 + m_2 g l_2 \cos(q_1 + q_2)$$

SEC 1.1.5: OBTAINING GIVEN SECOND-ORDER EQUATIONS OF MOTION:

We rearrange the equations 3,4 to obtain the second-order description of motion of the two-link manipulator.

We rearrange eq.3 to get the following:

$$((m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosq_2)\ddot{q_1}+(m_2l_2^2+m_2l_1l_2cosq_2)\ddot{q_2}+(-m_2l_1l_2\dot{q_2}sinq_2)\dot{q_1}+(-m_2l_1l_2(\dot{q_1}+\dot{q_2})sinq_2)\dot{q_2}$$
$$+(m_1+m_2)gl_1cosq_1+m_2gl_2\cos(q_1+q_2)=\tau_1$$

And we rearrange eq.4 to obtain the following:

$$(m_2l_2^2+m_2l_1l_2cosq_2)\ddot{q_1}+(m_2l_2^2)\ddot{q_2}+(m_2l_1l_2\dot{q_1}sinq_2)\dot{q_2}+m_2gl_2\cos(q_1+q_2)=\tau_2$$

Thus, now in matrix form, we have obtained:

$$\begin{bmatrix}((m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosq_2) & (m_2l_2^2+m_2l_1l_2cosq_2)\\ (m_2l_2^2+m_2l_1l_2cosq_2) & (m_2l_2^2)\end{bmatrix}\begin{bmatrix}\ddot{q_1}\\ \ddot{q_2}\end{bmatrix}$$
$$+\begin{bmatrix}(-m_2l_1l_2\dot{q_2}sinq_2) & (-m_2l_1l_2(\dot{q_1}+\dot{q_2})sinq_2)\\ 0 & (m_2l_1l_2\dot{q_1}sinq_2)\end{bmatrix}\begin{bmatrix}\dot{q_1}\\ \dot{q_2}\end{bmatrix} \qquad (eq5)$$
$$+\begin{bmatrix}(m_1+m_2)gl_1cosq_1+m_2gl_2\cos(q_1+q_2)\\ m_2gl_2\cos(q_1+q_2)\end{bmatrix}=\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix}$$

1.2: State-space modelling:

The obtained equation (eq.5) is expressed as a system of two non-linear 2nd-order differential equations in the following way.

We name the matrices and their elements in the following way:

$$\boldsymbol{M}(\boldsymbol{q})=\begin{bmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{bmatrix},\boldsymbol{q}=\begin{bmatrix}q_1\\ q_2\end{bmatrix},\boldsymbol{\tau}=\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix}$$

Matrix $\boldsymbol{M}$ denotes the inertia matrix where the individual matrix elements are:

$$M_{11}=((m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosq_2)$$

$$M_{12}=M_{21}=(m_2l_2^2+m_2l_1l_2cosq_2)$$

$$M_{22}=m_2l_2^2$$

Vector $\boldsymbol{q}$ denotes the joint vector which contains the angles q_1 and q_2 at joints as depicted in Figure 2, Also the input to the system is the torque vector $\boldsymbol{\tau}$ which contains the magnitudes of torques that we are applying to the 2-link manipulator system.

The Coriolis and centrifugal force matrix $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$:

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})=\begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}=\begin{bmatrix}(-m_2l_1l_2\dot{q_2}sinq_2) & (-m_2l_1l_2(\dot{q_1}+\dot{q_2})sinq_2)\\ 0 & (m_2l_1l_2\dot{q_1}sinq_2)\end{bmatrix}$$

And the Gravity Vector $\boldsymbol{G}(\boldsymbol{q})$:

$$\boldsymbol{G}(\boldsymbol{q})=\begin{bmatrix}G_{11}\\ G_{21}\end{bmatrix}=\begin{bmatrix}(m_1+m_2)gl_1cosq_1+m_2gl_2\cos(q_1+q_2)\\ m_2gl_2\cos(q_1+q_2)\end{bmatrix}$$

We observe that we have obtained the following equation of motion of the two-link manipulator:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{G}(\boldsymbol{q})=\boldsymbol{\tau} \qquad \ldots(eq.6)$$

1.2.1: OBTAINING STATE-SPACE EQUATION:

Now, for state-space formulation, firstly, we define the following 4 state variables:

$$u_1=q_1,u_2=\dot{q_1},u_3=q_2 \text{ and } u_4=\dot{q_2}$$

i.e, the state variable containing vector will be:

$$\begin{bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{bmatrix}=\begin{bmatrix}q_1\\ \dot{q_1}\\ q_2\\ \dot{q_2}\end{bmatrix}$$

Then, we derive the expression of $\ddot{q_1}$ and $\ddot{q_2}$ from eq.6 in the following way:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} + \begin{bmatrix} G_{11} \\ G_{21} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

Rearranging terms and multiplying with inverse of $\boldsymbol{M}(\boldsymbol{q})$ on both sides of equation,

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}^{-1}\left\{\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} - \begin{bmatrix} G_{11} \\ G_{21} \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}\right\}$$

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \frac{adj\left(\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\right)}{\det\left(\boldsymbol{M}(q)\right)}\left\{\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} - \begin{bmatrix} G_{11} \\ G_{21} \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}\right\}$$

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \frac{adj\left(\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\right)}{M_{22}M_{11} - (M_{21})^2}\left\{\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} - \begin{bmatrix} G_{11} \\ G_{21} \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}\right\} \qquad (since\ M_{12} = M_{21})$$

Adjoint of a 2x2 matrix is obtained directly by swapping diagonal elements and inverting the sign of the off-diagonal elements, using that, our equation becomes:

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \frac{1}{M_{22}M_{11} - (M_{21})^2}\begin{bmatrix} M_{22} & -M_{12} \\ -M_{21} & M_{11} \end{bmatrix}\left\{\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} - \begin{bmatrix} G_{11} \\ G_{21} \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}\right\}$$

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \frac{1}{M_{22}M_{11} - (M_{21})^2}\begin{bmatrix} M_{22} & -M_{12} \\ -M_{21} & M_{11} \end{bmatrix}\begin{bmatrix} \tau_1 - G_{11} - C_{11}\dot{q_1} - C_{12}\dot{q_2} \\ \tau_2 - G_{21} - C_{21}\dot{q_1} - C_{22}\dot{q_2} \end{bmatrix}$$

For simplifying our calculations, we take the following substitutions:

$$\text{Taking } \alpha = \tau_1 - G_{11} - C_{11}\dot{q_1} - C_{12}\dot{q_2}$$

$$\text{and } \beta = \tau_2 - G_{21} - C_{21}\dot{q_1} - C_{22}\dot{q_2}$$

Thus, we have:

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \frac{1}{M_{22}M_{11} - (M_{21})^2}\begin{bmatrix} \alpha M_{22} - \beta M_{12} \\ -\alpha M_{21} + \beta M_{11} \end{bmatrix}$$

Or,

$$\ddot{q_1} = \frac{\alpha M_{22} - \beta M_{12}}{M_{22}M_{11} - (M_{21})^2}$$

$$\ddot{q_2} = \frac{-\alpha M_{21} + \beta M_{11}}{M_{22}M_{11} - (M_{21})^2}$$

On substituting symbolic values, we get:

$$\det\left(\boldsymbol{M}(q)\right) = M_{22}M_{11} - (M_{21})^2 = m_2 l_1^2 l_2^2 (m_1 + m_2 sin^2 q_2)$$

$$\alpha = \tau_1 - (m_1 + m_2) g l_1 cos q_1 - m_2 g l_2 \cos(q_1 + q_2) + m_2 l_1 l_2 (2\dot{q_1} + \dot{q_2})\dot{q_2} sin q_2$$

$$\beta = \tau_2 - m_2 g l_2 \cos(q_1 + q_2) - m_2 l_1 l_2 \dot{q_1}\dot{q_2} sin q_2$$

$$\ddot{q_1} = \frac{(\tau_1 - \tau_2) l_2 - l_1 \tau_2 cos q_2 - (m_1 + m_2) g l_1 l_2 cos q_1 + m_2 l_1 l_2^2 (3\dot{q_1} + \dot{q_2})\dot{q_2} sin q_2 + m_2 l_1 l_2 g cos q_2 \cos(q_1 + q_2) + m_2 l_1^2 l_2 \dot{q_1}\dot{q_2} sin q_2 cos q_2}{l_1^2 l_2 (m_1 + m_2 sin^2 q_2)}$$

$$\ddot{q_2} = \frac{-m_2 l_2 (l_2 + l_1 cos q_2)\tau_1 + [(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 cos q_2]\tau_2 + m_2 (m_1 + m_2) l_1 l_2 (l_2 + l_1 cos q_2) g cos q_1 - [(m_1 + m_2) l_1^2 + m_2 l_1 l_2 cos q_2] m_2 g l_2 \cos(q_1 + q_2) - [m_2 l_2 (l_2 + l_1 cos q_2)] m_2 l_1 l_2 (2\dot{q_1} + \dot{q_2})\dot{q_2} sin q_2 - [(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 cos q_2] m_2 l_1 l_2 \dot{q_1}\dot{q_2} sin q_2}{m_2 l_1^2 l_2^2 (m_1 + m_2 sin^2 q_2)}$$

So, we now have our state-space equations as follows:

$$\begin{bmatrix} \dot{u_1} \\ \dot{u_2} \\ \dot{u_3} \\ \dot{u_4} \end{bmatrix} = \begin{bmatrix} u_2 \\ \frac{\tau_1 l_2 - (l_2 + l_1 cos u_3)\tau_2 - (m_1 + m_2) g l_1 l_2 cos u_1 + m_2 l_1 l_2^2 (3u_2 + u_4) u_4 sin u_3 + m_2 l_1 l_2 g cos u_3 \cos(u_1 + u_3) + m_2 l_1^2 l_2 u_2 u_4 sin u_3 cos u_3}{l_1^2 l_2 (m_1 + m_2 sin^2 u_3)} \\ u_4 \\ \frac{-m_2 l_2 (l_2 + l_1 cos u_3)\tau_1 + [(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 cos u_3]\tau_2 + m_2 (m_1 + m_2) l_1 l_2 (l_2 + l_1 cos u_3) g cos u_1 - [(m_1 + m_2) l_1^2 + m_2 l_1 l_2 cos u_3] m_2 g l_2 \cos(u_1 + u_3) - [m_2 l_2 (l_2 + l_1 cos u_3)] m_2 l_1 l_2 (2u_2 + u_4) u_4 sin u_3 - [(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 cos u_3] m_2 l_1 l_2 u_2 u_4 sin u_3}{m_2 l_1^2 l_2^2 (m_1 + m_2 sin^2 u_3)} \end{bmatrix}$$

State space form $\dot{\boldsymbol{X}}=\mathbf{AX}+\mathbf{BU}$

$$\begin{bmatrix} \dot{u_1} \\ \dot{u_2} \\ \dot{u_3} \\ \dot{u_4} \end{bmatrix} = \begin{bmatrix} u_2 \\ \frac{-(m_1+m_2)gl_1l_2cosu_1 + m_2l_1l_2^2(3u_2+u_4)u_4sinu_3 + m_2l_1l_2gcosu_3\cos(u_1+u_3) + m_2l_1^2l_2u_2u_4sinu_3cosu_3}{l_1^2l_2(m_1+m_2sin^2u_3)} \\ u_4 \\ \frac{m_2(m_1+m_2)l_1l_2(l_2+l_1cosu_3)gcosu_1 - [(m_1+m_2)l_1^2+m_2l_1l_2cosu_3]m_2gl_2\cos(u_1+u_3) - [m_2l_2(l_2+l_1cosu_3)]m_2l_1l_2(2u_2+u_4)u_4sinu_3 - [(m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosu_3]m_2l_1l_2u_2u_4sinu_3}{m_2l_1^2l_2^2(m_1+m_2sin^2u_3)} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 \\ l_2 & -(l_2+l_1cosu_3) \\ 0 & 0 \\ -m_2l_2(l_2+l_1cosu_3) & [(m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosu_3] \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

System matrix **A** can't be found out without Taylor's series expansion, as the expressions are non-linear and implicit as column vector **AX**.

$$AX = \begin{bmatrix} u_2 \\ \frac{-(m_1+m_2)gl_1l_2cosu_1 + m_2l_1l_2^2(3u_2+u_4)u_4sinu_3 + m_2l_1l_2gcosu_3\cos(u_1+u_3) + m_2l_1^2l_2u_2u_4sinu_3cosu_3}{l_1^2l_2(m_1+m_2sin^2u_3)} \\ u_4 \\ \frac{m_2(m_1+m_2)l_1l_2(l_2+l_1cosu_3)gcosu_1 - [(m_1+m_2)l_1^2+m_2l_1l_2cosu_3]m_2gl_2\cos(u_1+u_3) - [m_2l_2(l_2+l_1cosu_3)]m_2l_1l_2(2u_2+u_4)u_4sinu_3 - [(m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosu_3]m_2l_1l_2u_2u_4sinu_3}{m_2l_1^2l_2^2(m_1+m_2sin^2u_3)} \end{bmatrix}$$

Input Matrix B:

$$B = \begin{bmatrix} 0 & 0 \\ l_2 & -(l_2+l_1cosu_3) \\ 0 & 0 \\ -m_2l_2(l_2+l_1cosu_3) & [(m_1+m_2)l_1^2+m_2l_2^2+2m_2l_1l_2cosu_3] \end{bmatrix}$$

Input to the system:

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

The initial conditions of the system is denoted by the vector:

$$X_0 = \begin{bmatrix} u_1(0) \\ u_2(0) \\ u_3(0) \\ u_4(0) \end{bmatrix} = \begin{bmatrix} q_1(0) \\ \dot{q_1}(0) \\ q_2(0) \\ \dot{q_2}(0) \end{bmatrix}$$

The error e that we'll pass to the controller will be:

The actual angles that are obtained when the input torques are applied to the system are denoted by the following:

$$\boldsymbol{q_{ac}} = \begin{bmatrix} q_{1,actual} \\ q_{2,actual} \end{bmatrix}$$

Thus, the error that we'll be passing to the controller will be:

$$e = q - q_{ac} = \begin{bmatrix} q_1 - q_{1,actual} \\ q_2 - q_{2,actual} \end{bmatrix}$$

2. Controller Design for the 2-Link Manipulator

In a two-link robotic manipulator, the motion of each joint is governed by nonlinear and coupled dynamics. Without proper control, the system may exhibit unstable or inaccurate behaviour. Therefore, controllers play an essential role in ensuring precise and reliable operation. Their significance can be summarized as follows:

- **Ensuring Precision and Accuracy**

Controllers minimize the deviation between the desired and actual joint positions. This is crucial in applications such as welding, assembly, or material handling, where even minor angular errors can lead to task failure.

- **Smooth Trajectory Tracking**

By continuously comparing real-time joint angles with the reference trajectory, controllers generate appropriate corrective torques, ensuring that the manipulator moves along the desired path without jerks or oscillations.

- **Adaptive Response to Dynamic Conditions**

External disturbances such as sudden payload changes or unexpected contacts can affect stability. A well-tuned controller adjusts the control effort instantaneously to maintain consistent motion.

- **Real-Time Error Detection and Compensation**

Through feedback, the controller monitors tracking error and actively compensates for it. This allows the manipulator to correct deviations as they occur, improving reliability during continuous operation.

- **Regulation of Interaction Forces**

In tasks involving physical contact (e.g., pressing, gripping, or polishing), controllers regulate the actuator outputs to prevent excessive force, ensuring safety for both the manipulator and the environment.

- **Management of Complex Dynamics**

The coupled dynamics of a two-link system can be difficult to handle manually. Controllers simplify this by automatically generating appropriate torques based on feedback, allowing higher-level tasks to be executed without manual tuning.

- **Facilitating Simulation and Tuning**

Before implementing on hardware, control strategies can be tested in simulation using the controller structure. This helps evaluate system performance, stability, and settling characteristics without physical risk.

For this project, the primary objective is to design controllers that drive the manipulator from its initial configuration of $q(0) = [0.2, 0.15]^T$ radians to the desired configuration of $q_{desired} = [0,0]^T$ radians. The controller's role is to calculate the necessary torques (τ) to apply to the joints to achieve this goal efficiently and accurately.

We will implement and analyze three control strategies based on Proportional (P), Integral (I), and Derivative (D) actions: a PD, PI, and PID controller. The effect of each component on the manipulator's performance is as follows:

- **Proportional (P) Control:** This component generates a control torque that is directly proportional to the current error, which is the difference between the desired and actual joint angles ($e = q_{desired} - q$). It acts like a virtual spring, pulling the manipulator links toward their target positions. A higher proportional gain results in a faster response but can lead to instability and overshoot.
- **Derivative (D) Control:** This component predicts the future error by looking at the error's rate of change. It provides a damping effect, counteracting the manipulator's velocity as it approaches the target. In this project, the D-action is crucial for reducing overshoot and improving the settling time, making the system more stable.

- **Integral (I) Control:** This component works to eliminate any persistent error by accumulating it over time. Its main purpose is to ensure high steady-state accuracy, driving the final error to zero. This is essential for overcoming constant disturbances like friction or slight modelling inaccuracies, ensuring the manipulator stops precisely at the desired [0, 0] configuration.

2.1 Prerequisites: Dynamic Decoupling of the Manipulator

For control design, we first express the joint-space dynamics of the two-link manipulator as:

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

where

- $M(q)$ is the inertia matrix
- $C(q,\dot{q})$ is the Coriolis/centrifugal matrix
- $G(q)$ represents gravitational torques
- $\tau_b = M^{-1}(q)\,\tau$ is the *virtual input* used for control design

This transformation **decouples the nonlinear dynamics**, allowing us to define a **new control input**:

$$\tau_b = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad \Rightarrow \qquad \tau = M(q) \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Define the tracking error for each joint as:

$$e_1 = q_{1,ref} - q_1, \qquad e_2 = q_{2,ref} - q_2$$

2.2 Proportional-Derivative (PD) Control

A PD controller generates corrective input based on **instantaneous error and its rate of change**:

$$f = K_p\,e + K_d\,\frac{de}{dt}$$

Applying Laplace Transform:

$$F(s) = K_p E(s) + K_d\,s\,E(s)$$

For each joint:

$$\begin{aligned} f_1 &= K_p(q_{1,ref} - q_1) + K_d(\dot{q}_{1,ref} - \dot{q}_1) \\ f_2 &= K_p(q_{2,ref} - q_2) + K_d(\dot{q}_{2,ref} - \dot{q}_2) \end{aligned}$$

Effect:

- Proportional term reduces **instantaneous error**
- Derivative term provides **damping**, preventing oscillations
- However, PD **cannot eliminate steady-state error** under gravity or disturbance

2.3 Proportional-Integral (PI) Control

To eliminate long-term offset, the integral of error is added:

$$f = K_p e + K_i \int e\, dt$$

Laplace Transform:

$$F(s) = K_p E(s) + \frac{K_i}{s} E(s)$$

Joint-wise:

$$\begin{aligned} f_1 &= K_p(q_{1,ref} - q_1) + K_i \int (q_{1,ref} - q_1)\, dt \\ f_2 &= K_p(q_{2,ref} - q_2) + K_i \int (q_{2,ref} - q_2)\, dt \end{aligned}$$

Effect:

- Integral term drives **steady-state error to zero**
- But may introduce **slower response or oscillation** without damping

2.4 Proportional-Integral-Derivative(PID) Control

Combining all three actions:

$$f = K_p e + K_d \frac{de}{dt} + K_i \int e\, dt$$

Laplace Transform:

$$F(s) = K_p E(s) + K_d s E(s) + \frac{K_i}{s} E(s)$$

Thus:

$$\begin{aligned} f_1 &= K_p(q_{1,ref} - q_1) + K_d(\dot{q}_{1,ref} - \dot{q}_1) + K_i \int (q_{1,ref} - q_1)\, dt \\ f_2 &= K_p(q_{2,ref} - q_2) + K_d(\dot{q}_{2,ref} - \dot{q}_2) + K_i \int (q_{2,ref} - q_2)\, dt \end{aligned}$$

3 The TwoLink_PID_Dynamics Code

```
function dqdt = TwoLink_PID_Dynamics(t,q,m1,m2,l1,l2,g,Kp,Ki,Kd, …
                          q_desired, e_int_prev, tspan)
   q1  = q(1);
   q2  = q(2);
   dq1 = q(3);
   dq2 = q(4);

  M11 = (m1 + m2)*l1^2 + m2*l2*(l2 + 2*l1*cos(q2));
  M12 = m2*l2*(l2 + l1*cos(q2));
  M21=M12;
  M22 = m2*l2^2;
  M = [M11 , M12 ; M21 , M22];

  C11 = -m2*l1*l2*sin(q2)*dq2 ;
  C12 = -m2*l1*l2*sin(q2)*(dq1+dq2);
  C21 = 0 ;
  C22=-C11;
  C = [C11 , C12 ; C21 , C22];

  G11 = m1*l1*g*cos(q1) + m2*g*(l2*cos(q1+q2) + l1*cos(q1));
  G21 = m2*g*l2*cos(q1+q2);
  G = [G11;G21];

  e = q_desired - [q1;q2];
  e_int = e_int_prev + e * (t - tspan(1));


  a1 = Kp(1) * (q_desired(1) - q1) - Kd(1) * dq1 + Ki(1) * e_int(1);
  a2 = Kp(2) * (q_desired(2) - q2) - Kd(2) * dq2 + Ki(2)* e_int(2);

  tau = (M) * [a1;a2];

  ddq = (tau - (C * [dq1; dq2]) - G) ./ M;
  dqdt = [dq1; dq2; ddq(1); ddq(2)];

end
```

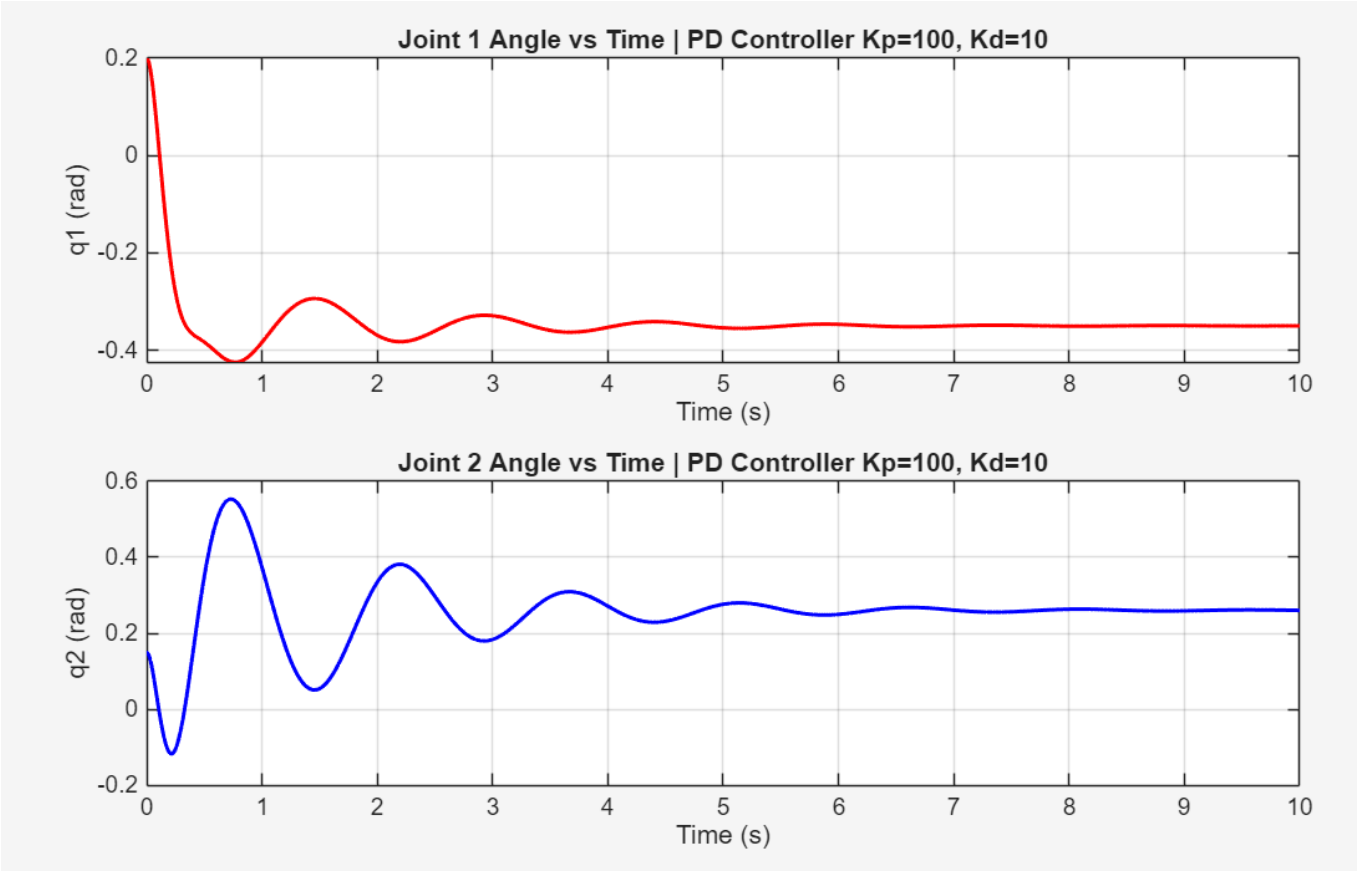
3.1 Proportional Derivative Controller

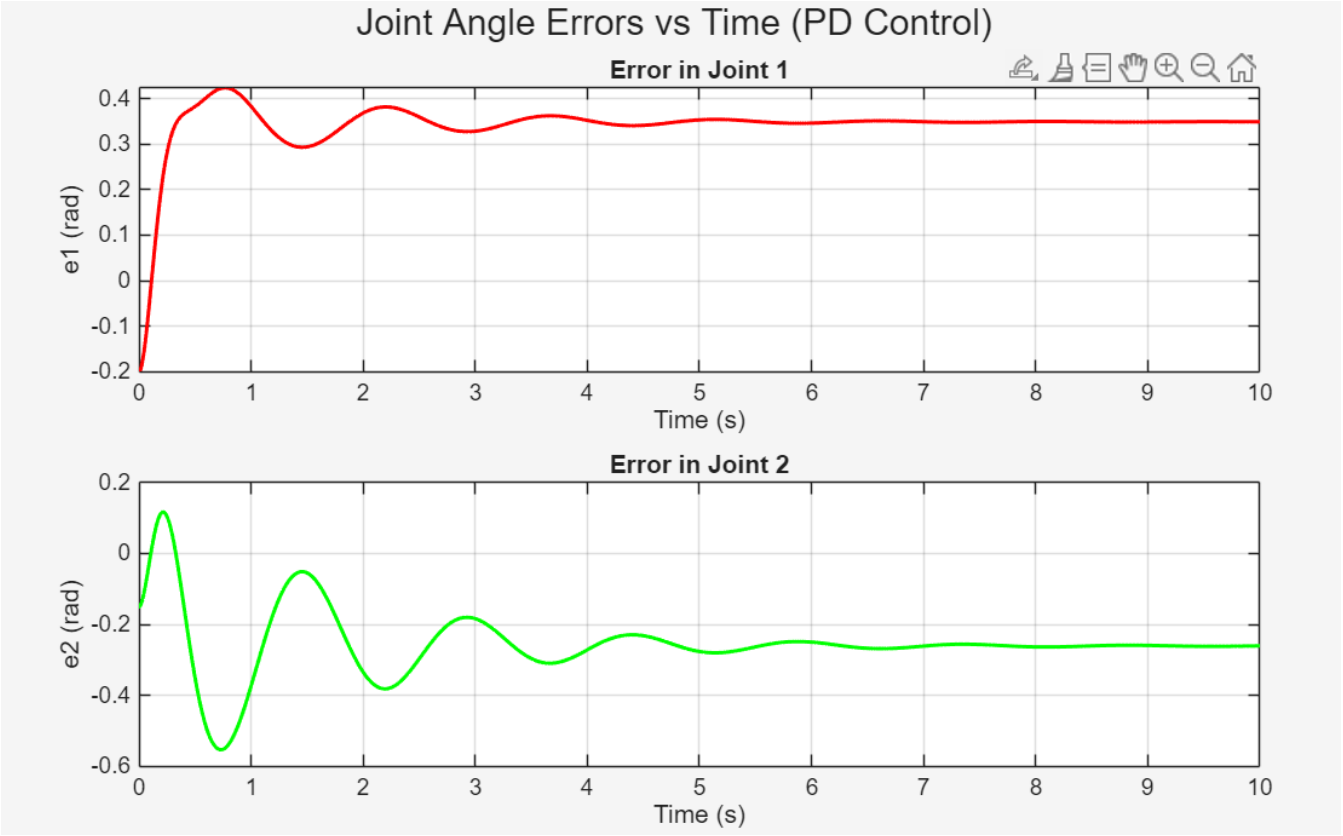
3.1.1 Code

```
% Two-Link Manipulator PD Simulation

clc; clear; close all;
m1 = 5;
m2 = 3;
l1 = 0.25;
l2 = 0.15;
g  = 9.81;
kp = 100; kd = 10; ki = 0;
Kp = [kp, kp];
Kd = [kd, kd];
Ki = [ki, ki];

q_initial = [0.2; 0.15; 0; 0];
q_desired = [0; 0];
e_int_prev = [0; 0];

tspan = [0 10];

options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t, q] = ode45(@(t, q) TwoLink_PID_Dynamics(t, q, m1, m2, l1, l2, g,
Kp, Ki, Kd, …
                q_desired, e_int_prev, tspan), tspan, q_initial,
options);
q1 = q(:,1);
q2 = q(:,2);

figure;
subplot(2,1,1);
plot(t, q1, 'r', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('q1 (rad)');
title(['Joint 1 Angle vs Time | PD Controller Kp=' num2str(kp) ', Kd='
num2str(kd)]);
subplot(2,1,2);
plot(t, q2, 'b', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('q2 (rad)');
title(['Joint 2 Angle vs Time | PD Controller Kp=' num2str(kp) ', Kd='
num2str(kd)]);

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

figure;
subplot(2,1,1);
plot(t, e1, 'r', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('e1 (rad)');
title('Error in Joint 1'); subplot(2,1,2);
plot(t, e2, 'g', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('e2 (rad)');
title('Error in Joint 2');
sgtitle('Joint Angle Errors vs Time (PD Control)');
```

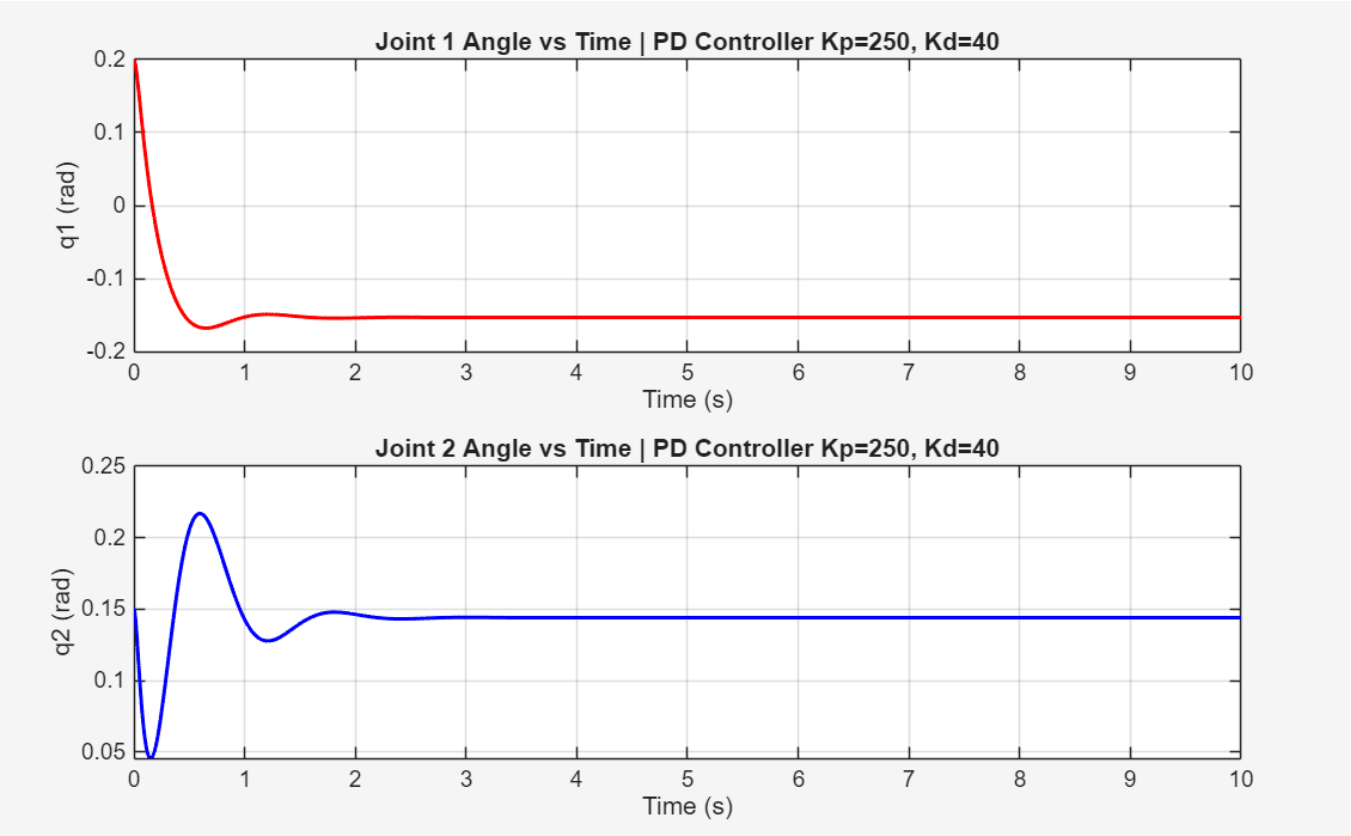
3.1.2 Plots

a) Kp=100, Kd=10





b) Kp=250, Kd=40


Joint 1 Angle vs Time | PD Controller Kp=250, Kd=40
q1 (rad)
Time (s)
Joint 2 Angle vs Time | PD Controller Kp=250, Kd=40
q2 (rad)
Time (s)


Joint Angle Errors vs Time (PD Control)
Error in Joint 1
e1 (rad)
Time (s)
Error in Joint 2
e2 (rad)
Time (s)

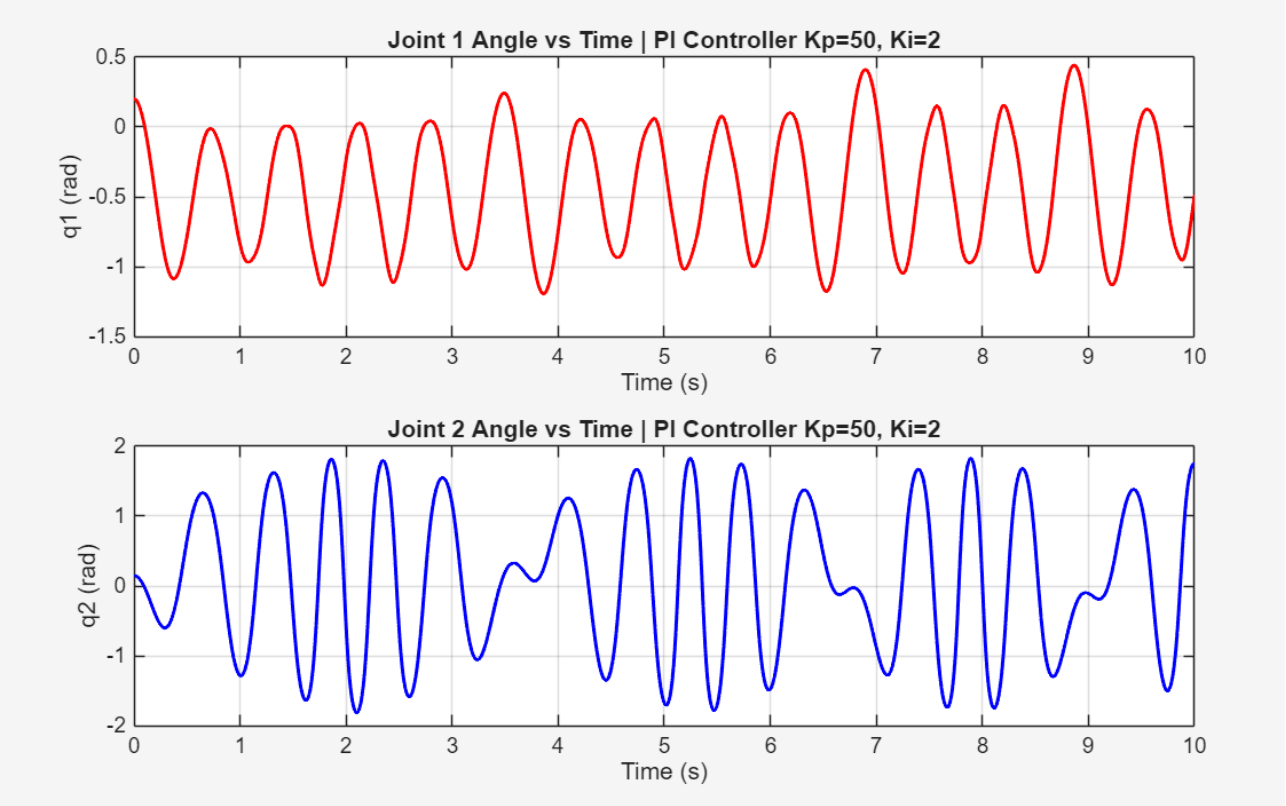
3.2 Proportional Integral Controller

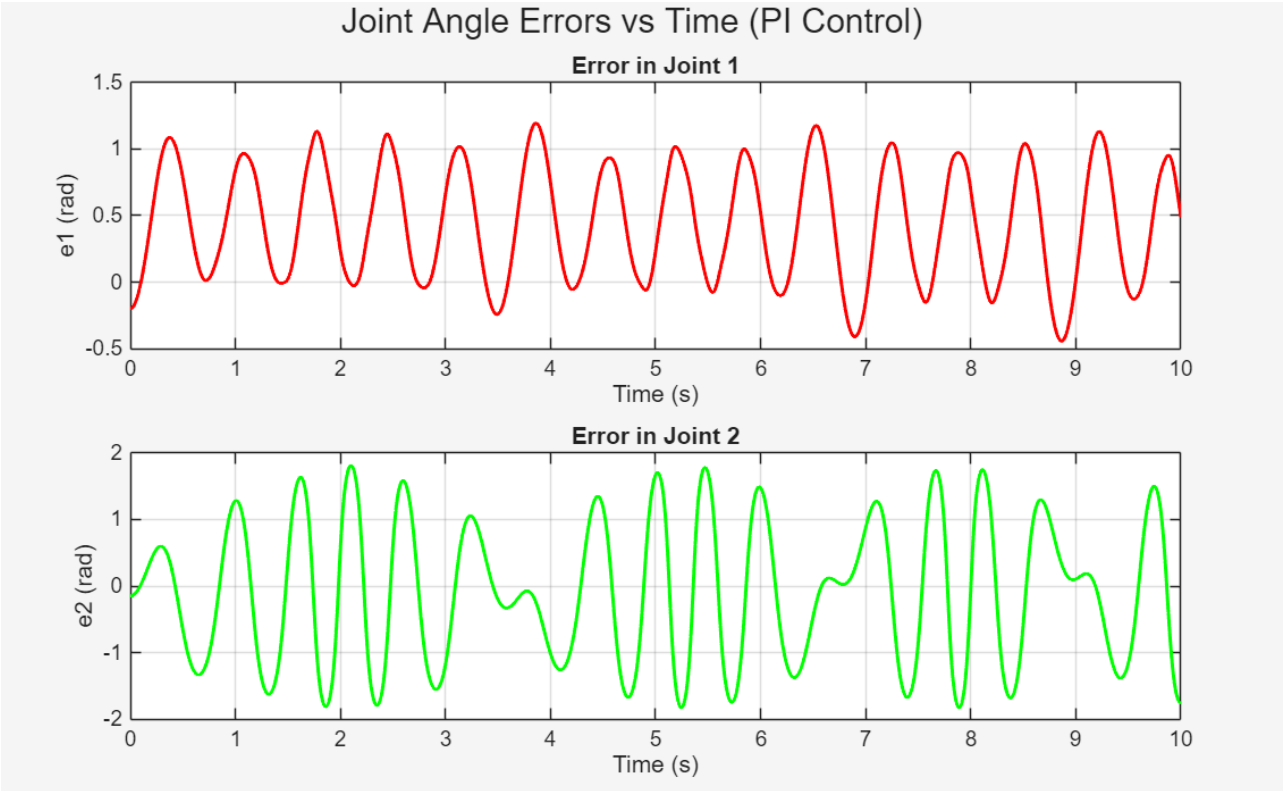
3.2.1 Code

```
% Two-Link Manipulator PI Simulation
clc; clear; close all;
m1 = 5;
m2 = 3;
l1 = 0.25;
l2 = 0.15;
g  = 9.81;

kp = 50; kd = 0; ki = 2;
Kp = [kp, kp];
Kd = [kd, kd];
Ki = [ki, ki];

q_initial = [0.2; 0.15; 0; 0];
q_desired = [0; 0];
e_int_prev = [0; 0];
tspan = [0 10];

options = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.01); % smoother
PI simulation
[t, q] = ode45(@(t, q) TwoLink_PID_Dynamics(t, q, m1, m2, l1, l2, g,
Kp, Ki, Kd, …
                q_desired, e_int_prev, tspan), tspan, q_initial,
options);
q1 = q(:,1);
q2 = q(:,2);

figure;
subplot(2,1,1);
plot(t, q1, 'r', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('q1 (rad)');
title(['Joint 1 Angle vs Time | PI Controller Kp=' num2str(kp) ', Ki='
num2str(ki)]);
subplot(2,1,2);
plot(t, q2, 'b', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('q2 (rad)');
title(['Joint 2 Angle vs Time | PI Controller Kp=' num2str(kp) ', Ki='
num2str(ki)]);
e1 = q_desired(1) - q1;
e2 = q_desired(2) - q2;
figure;
subplot(2,1,1);
plot(t, e1, 'r', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('e1 (rad)');
title('Error in Joint 1'); subplot(2,1,2);
plot(t, e2, 'g', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('e2 (rad)');
title('Error in Joint 2');
sgtitle('Joint Angle Errors vs Time (PI Control)');
```

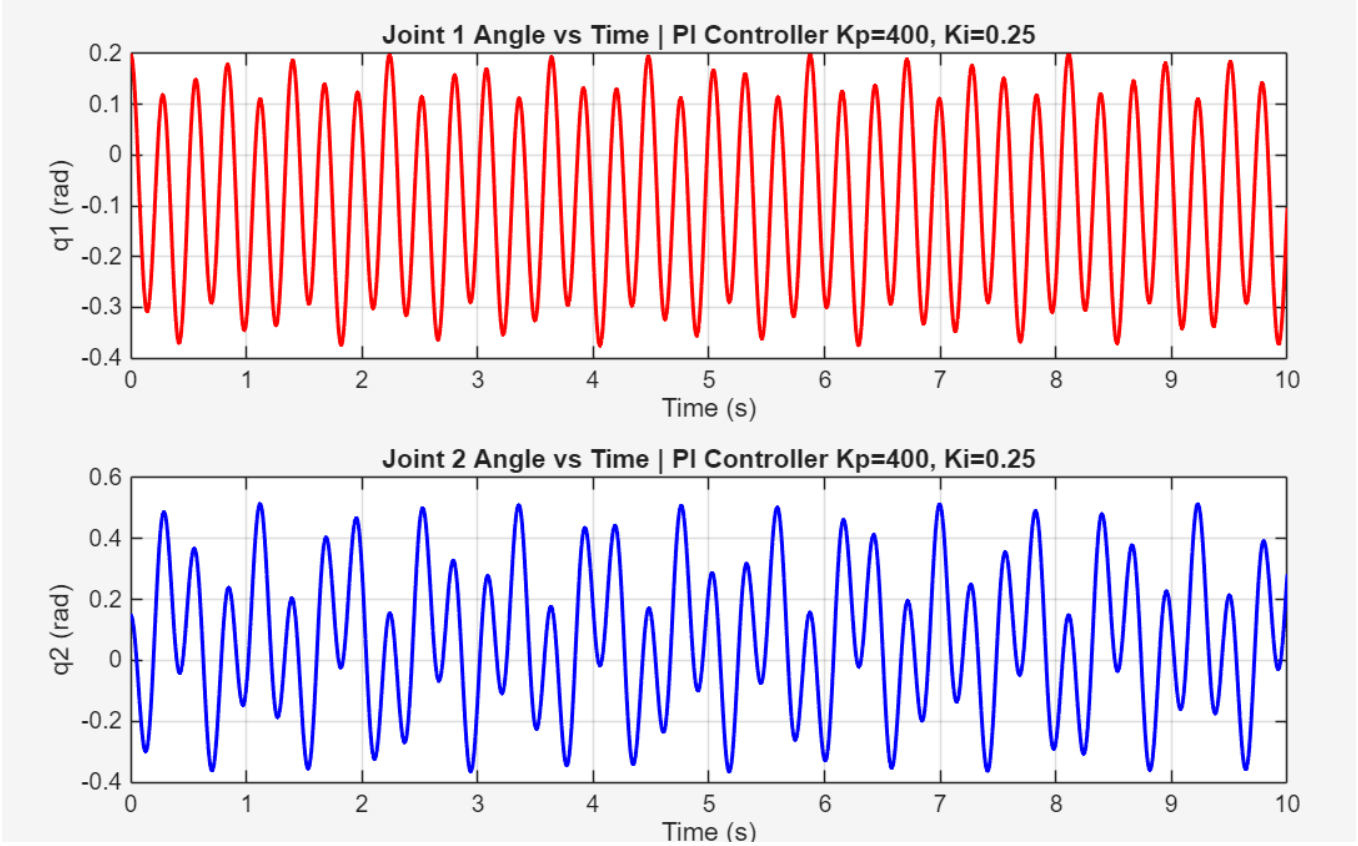
3.2.2 Plots

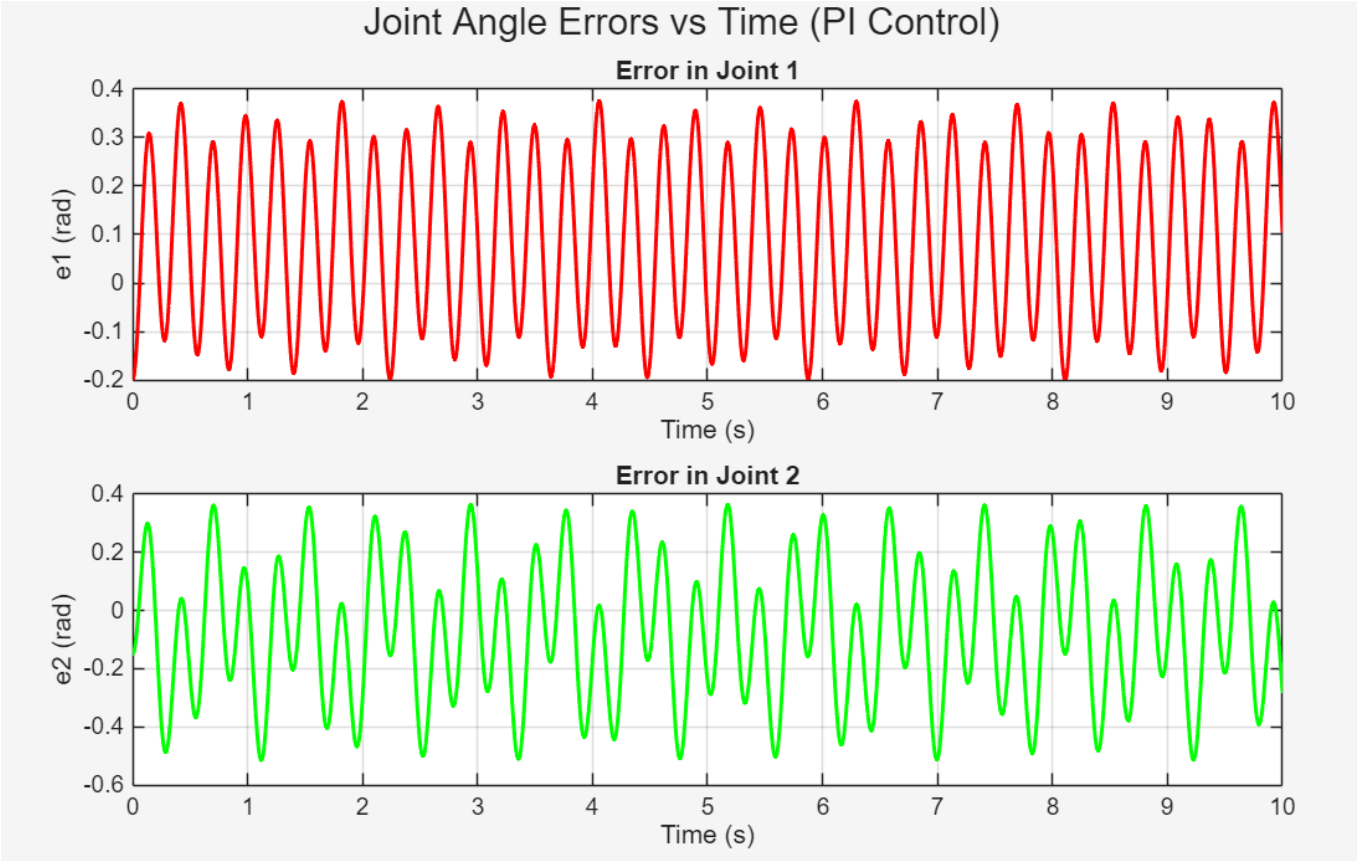
a) Kp=50 Ki=2



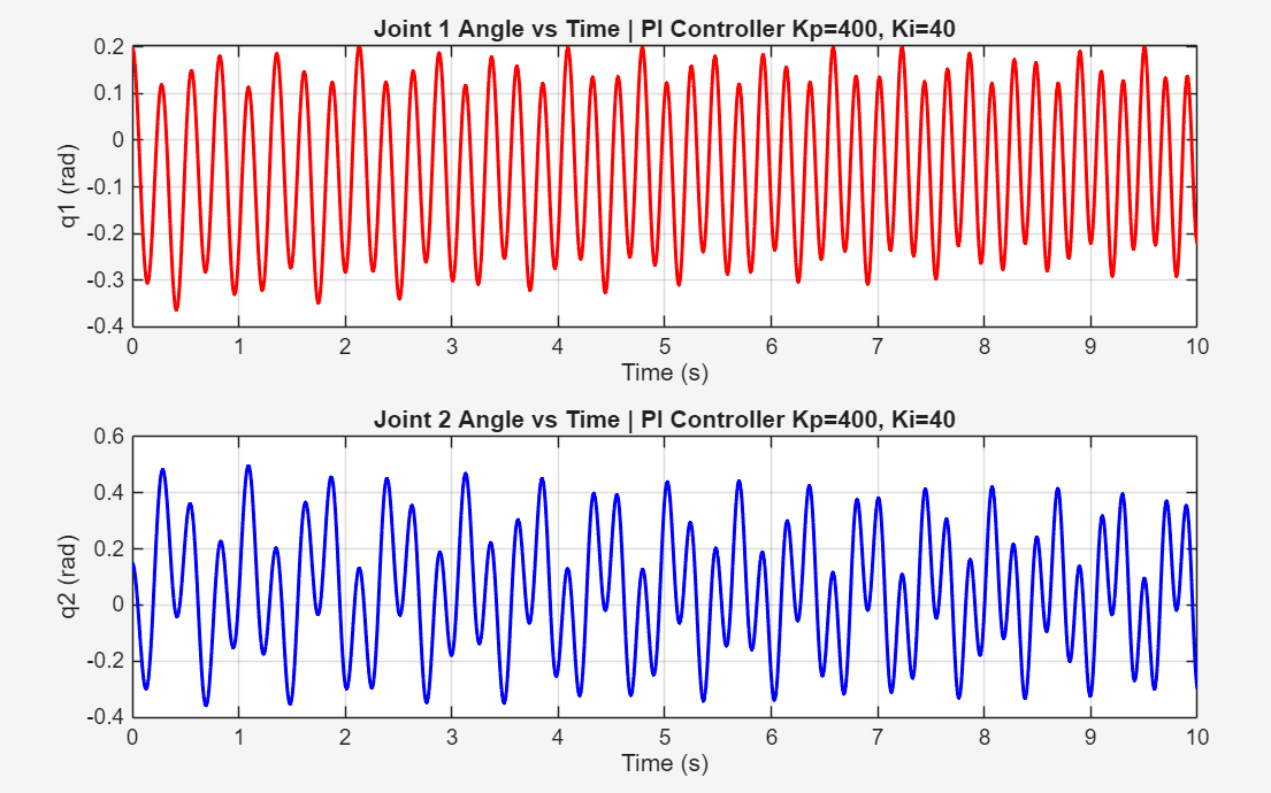


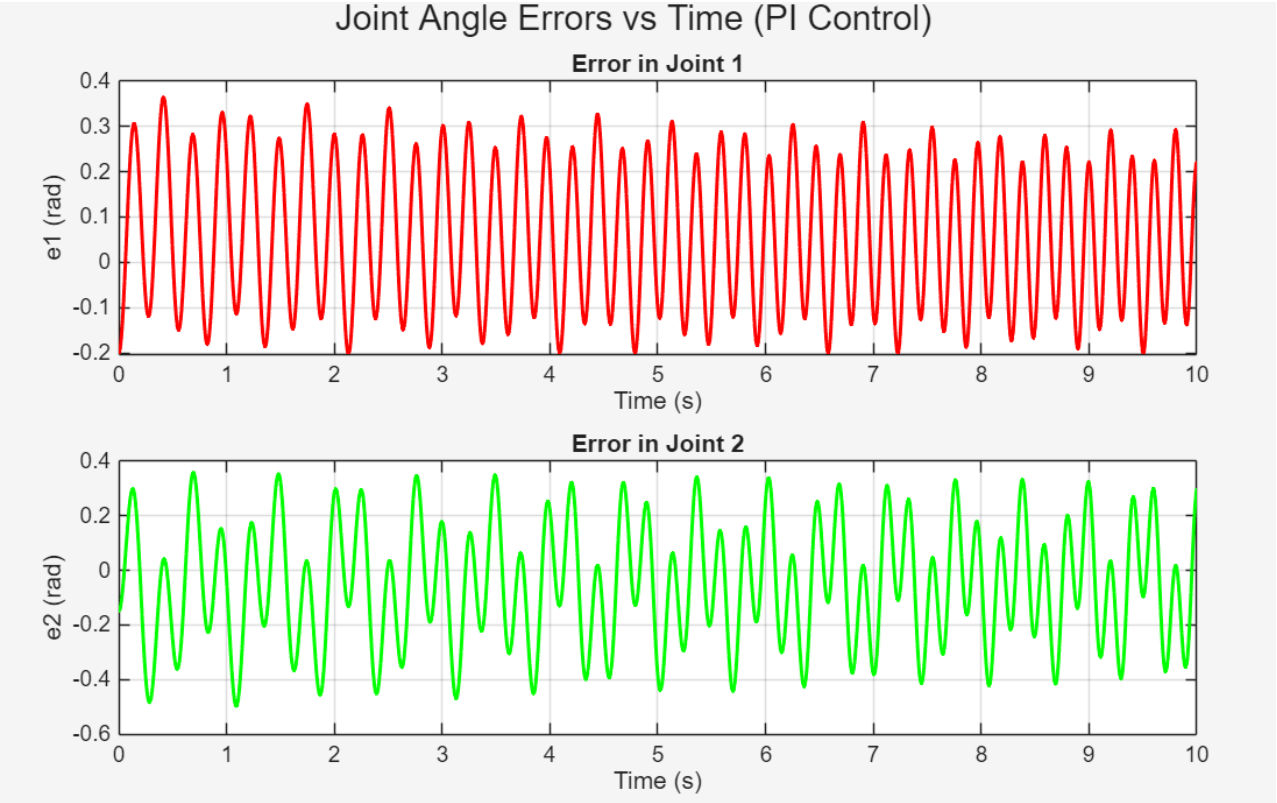
b) Kp=400 Ki=0.25





c) Kp=400 Ki=40


Joint 1 Angle vs Time | PI Controller Kp=400, Ki=40
q1 (rad)
Time (s)
Joint 2 Angle vs Time | PI Controller Kp=400, Ki=40
q2 (rad)
Time (s)


Joint Angle Errors vs Time (PI Control)
Error in Joint 1
e1 (rad)
Time (s)
Error in Joint 2
e2 (rad)
Time (s)

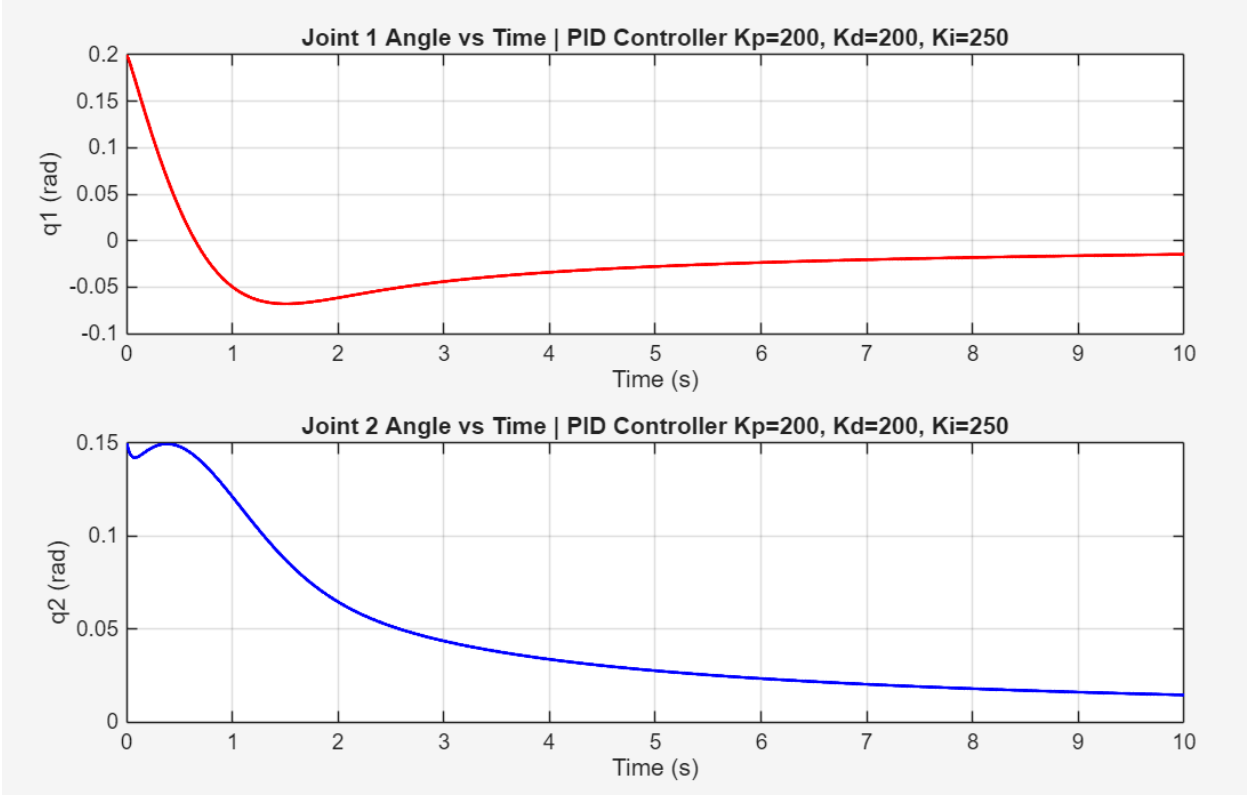
3.3 Proportional Integral Derivative Controller

3.3.1 Code

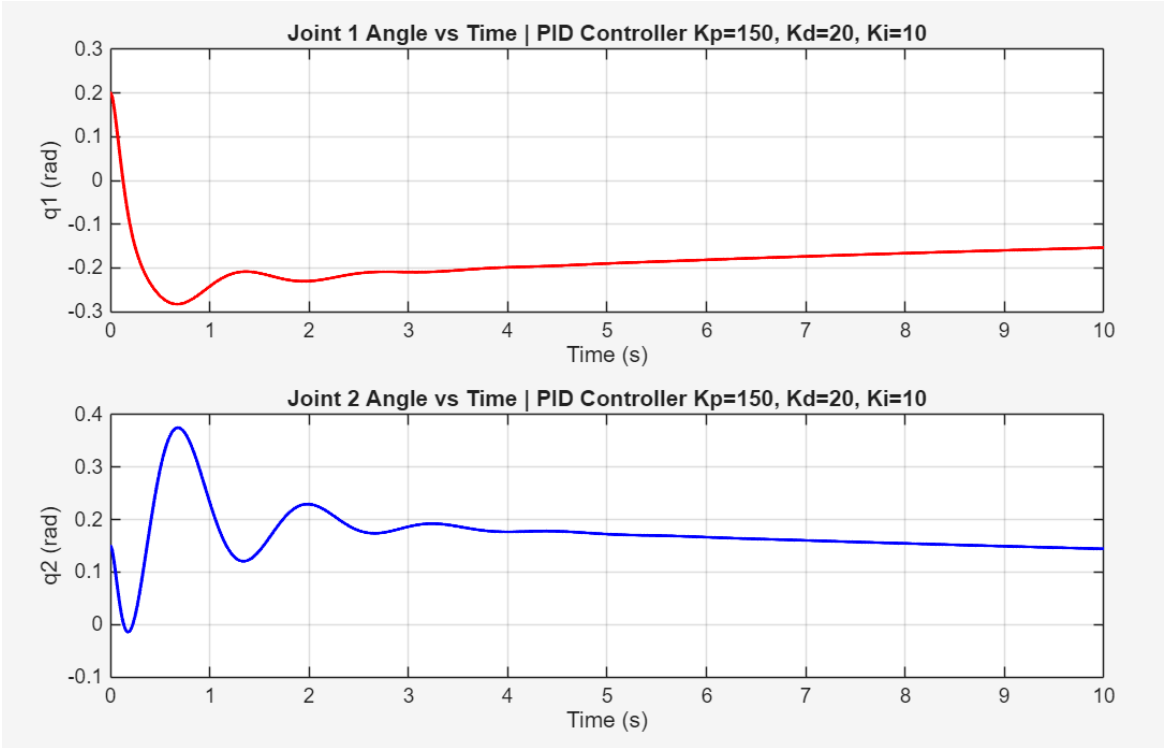
```
% Two-Link Manipulator PID Simulation

clc; clear; close all;
m1 = 5;
m2 = 3;
l1 = 0.25;
l2 = 0.15;
g  = 9.81;

kp = 200; kd = 200; ki = 250;
Kp = [kp, kp];
Kd = [kd, kd];
Ki = [ki, ki];

q_initial = [0.2; 0.15; 0; 0];
q_desired = [0; 0];
e_int_prev = [0; 0];
tspan = [0 10];
options = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.01);
[t, q] = ode45(@(t, q) TwoLink_PID_Dynamics(t, q, m1, m2, l1, l2, g,
Kp, Ki, Kd, …
                 q_desired, e_int_prev, tspan), tspan, q_initial,
options);
q1 = q(:,1);
q2 = q(:,2);
figure;
subplot(2,1,1);
plot(t, q1, 'r', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('q1 (rad)');
title(['Joint 1 Angle vs Time | PID Controller Kp=' num2str(kp) ', Kd='
num2str(kd) ', Ki=' num2str(ki)]);
subplot(2,1,2);
plot(t, q2, 'b', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('q2 (rad)');
title(['Joint 2 Angle vs Time | PID Controller Kp=' num2str(kp) ', Kd='
num2str(kd) ', Ki=' num2str(ki)]);
e1 = q_desired(1) - q1;
e2 = q_desired(2) - q2;
figure;
subplot(2,1,1);
plot(t, e1, 'r', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('e1 (rad)');
title('Error in Joint 1'); subplot(2,1,2);
plot(t, e2, 'g', 'LineWidth', 1.5); grid on;
xlabel('Time (s)'); ylabel('e2 (rad)');
title('Error in Joint 2');
sgtitle('Joint Angle Errors vs Time (PID Control)');
```

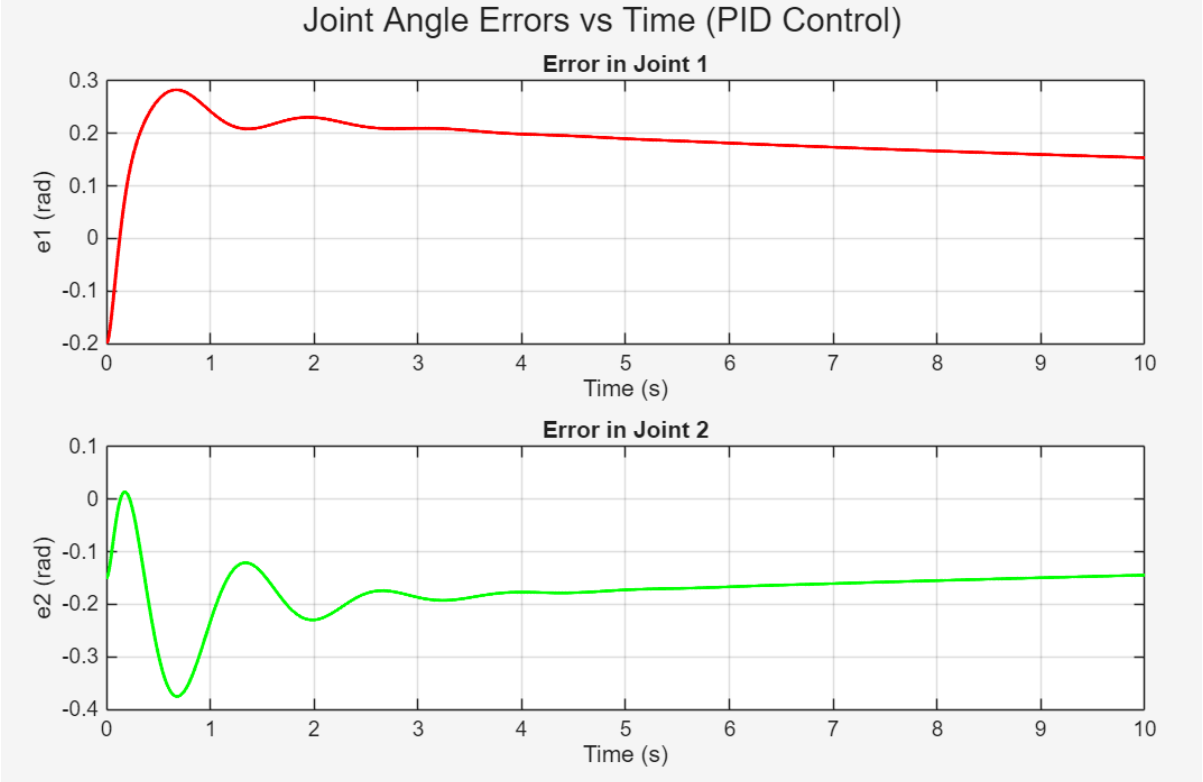
3.3.2 Plots

a) Kp=200 Kd=200 Ki=250



b) Kp=150 Kd=20 Ki=10





4 SIMULINK

a) System Configuration and Control

- The fundamental objective of the system is to achieve a specific, desired orientation
- This target is formally established by a reference signal, which provides the precise final angles the system is intended to reach
- A controller is implemented as the system's decision-making core to intelligently guide the system towards this reference goal
- The controller continuously computes an output command based on two critical inputs:
    - The system's unchanging physical constants (such as mass, length, or inertia)
    - A dynamically calculated error value
- The crucial error signal is generated by a subtractor block that constantly measures the discrepancy between the desired reference angles and the system's actual angular position at any given moment

b) State-Space Dynamics

- The physical laws governing the system's motion are mathematically modeled and encapsulated within a state-space subsystem block
- This block serves as a digital twin of the real-world system, predicting its behavior
- It takes the calculated output from the controller and combines it with a full snapshot of the system's current states:
    - Present angles
    - Angular velocities (first derivatives)

- Within this subsystem, the model executes the system's equations of motion to determine the instantaneous angular accelerations (second-order derivative terms)
- This calculation is a direct function of:
    - The controller's command
    - Current angular position
    - Current angular velocity

c) **Integration and State Calculation**

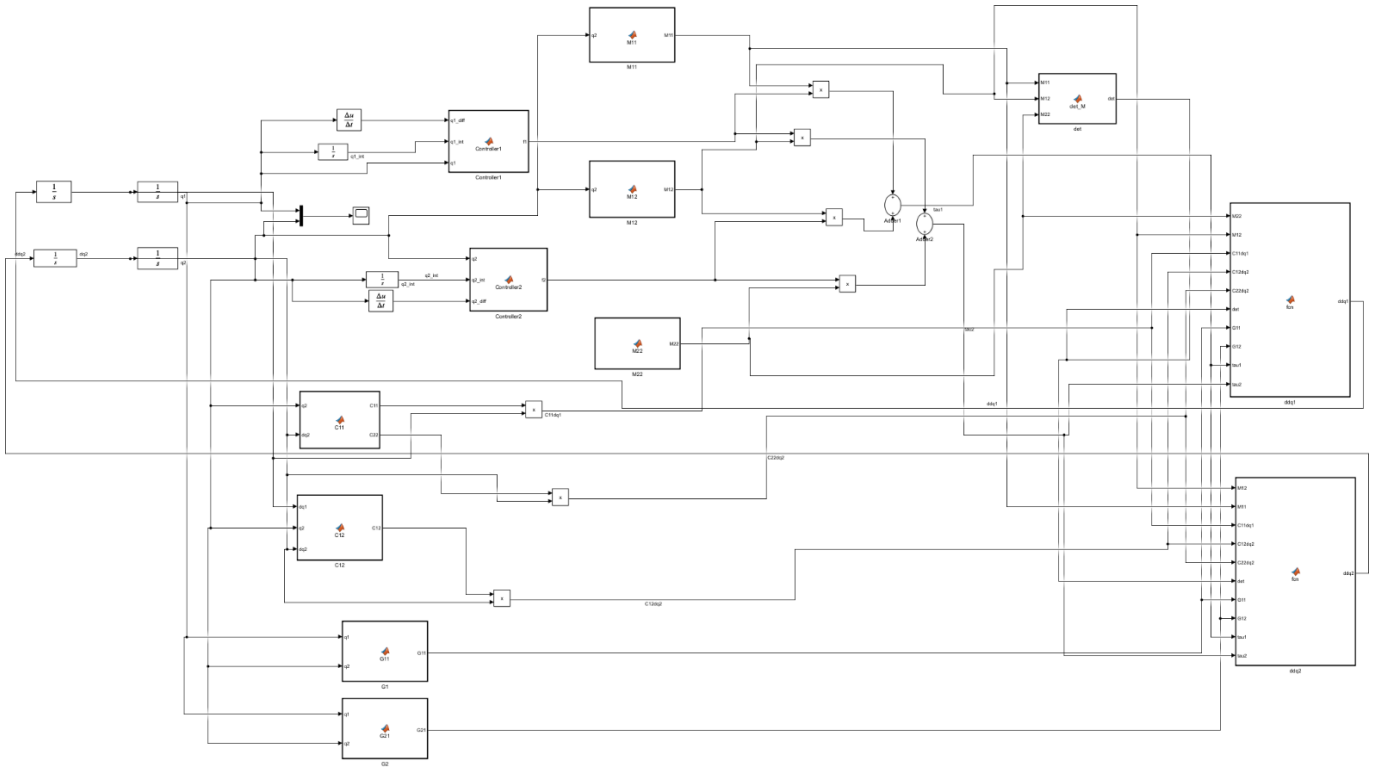
- The second-order derivatives are processed through a cascade of integrator blocks to translate calculated accelerations into motion over time
- First stage of integration:
    - Takes angular acceleration as input
    - Yields the first-order derivatives (angular velocities)
    - Determines the system's current angular velocities by accumulating the effects of acceleration over time
- Second stage of integration:
    - Takes angular velocities as input
    - Calculates the system's angles
    - Determines the new angular position based on the system's velocity
- The entire simulated trajectory begins from a known starting point by setting the initial conditions for the angles within the appropriate integrator block before the simulation begins Simulation

d) Loop and Progression

- The entire implementation operates as a continuous closed-loop feedback system

- The newly calculated angles are immediately fed back to the beginning of the process
- They are compared against the reference signal to generate a fresh, updated error value for the controller
- The simulation progresses iteratively, advancing in discrete time steps
- At the conclusion of each step, the newly computed state values (both angles and their derivatives) serve as the inputs for the next cycle of calculations
- This repetitive cycle continues relentlessly:
    - Error calculation
    - Controller action
    - Dynamic modeling
    - Integration
- The simulation launches from the defined initial conditions and runs until the total specified time duration has been reached, allowing the model to simulate the system's dynamic response over time

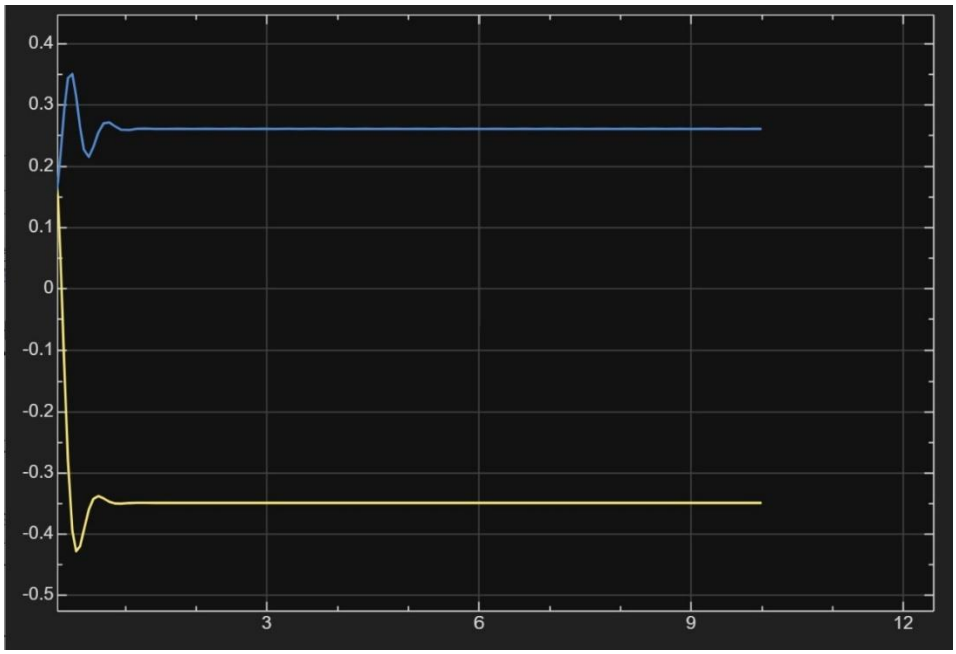
4.1 BLOCK DIAGRAM



4.2 PROPORTION DERIVATIVE CONTROLLER

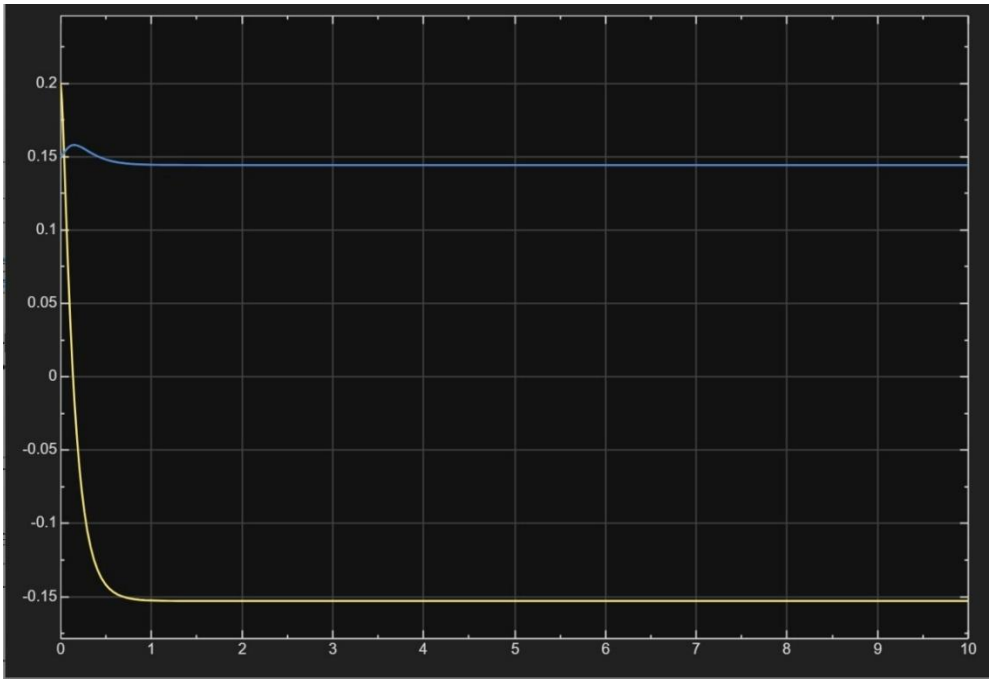
(Blue is q2 and yellow q1)

Kp1 = 100, Kd1 = 10
Kp2 = 100, kd2 = 10



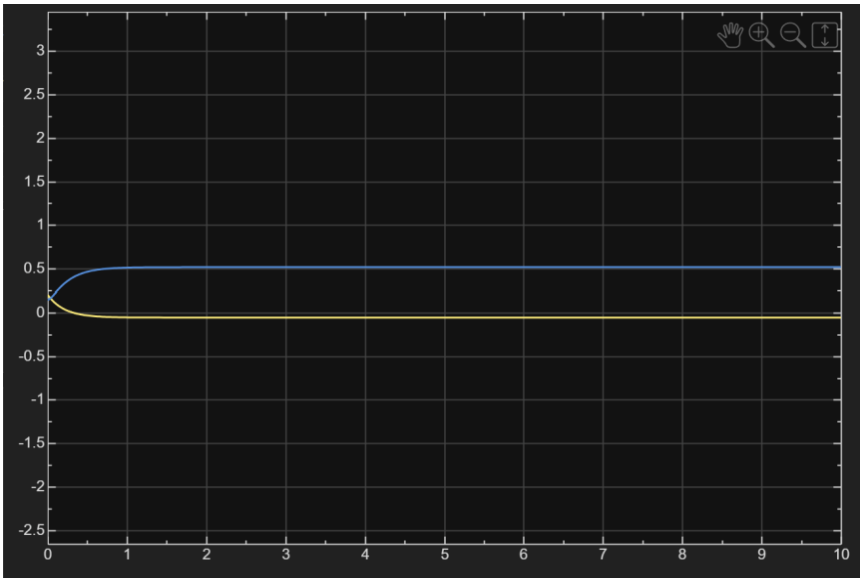
Kp1 = 250, Kd1 = 40

Kp2 = 250, kd2 = 40



Kp1 = 700, Kd1 = 150

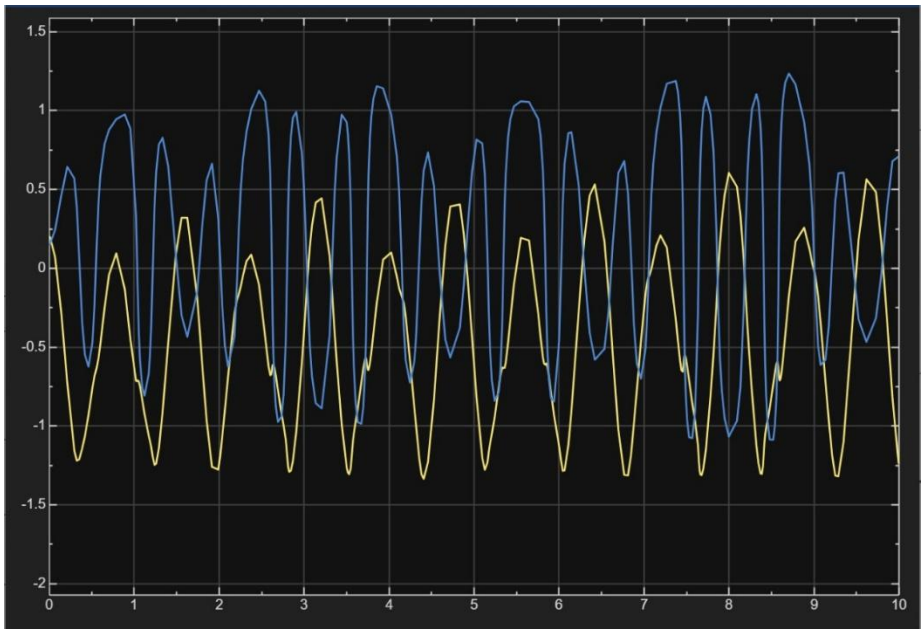
Kp2 = 70, kd2 = 25



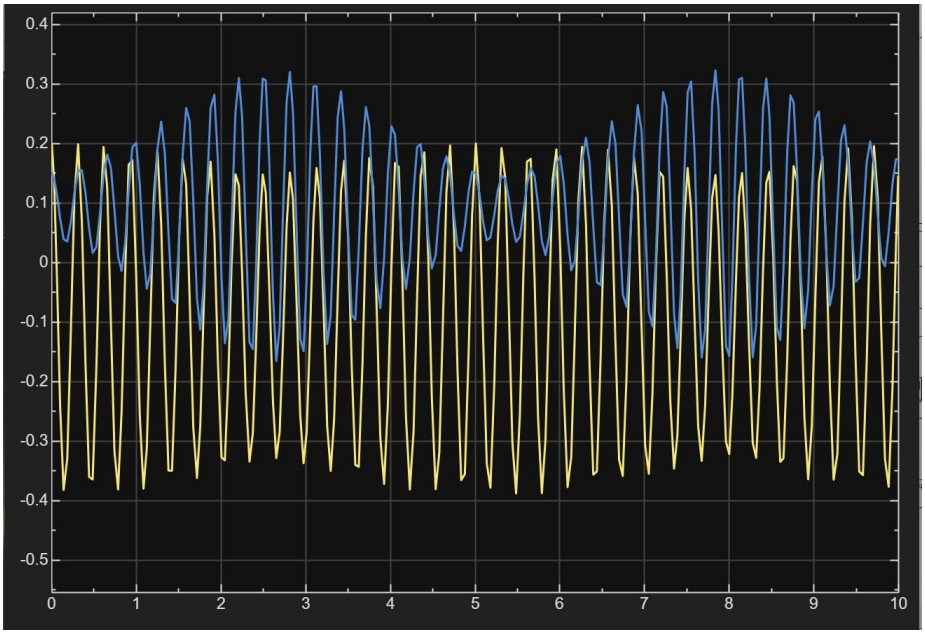
4.3 PROPORTION INTEGRAL CONTROLLER

Kp1 = 50, Ki1 = 2

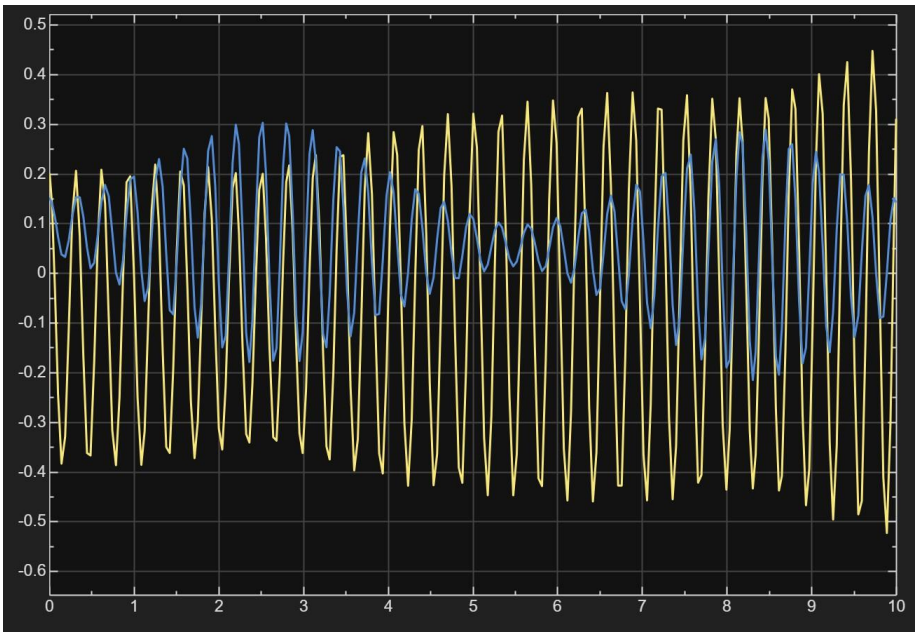
Kp2 = 50, Ki2 = 2



Kp1 = 400, Ki1 = 0.25
Kp2 = 400, ki2 = 0.25

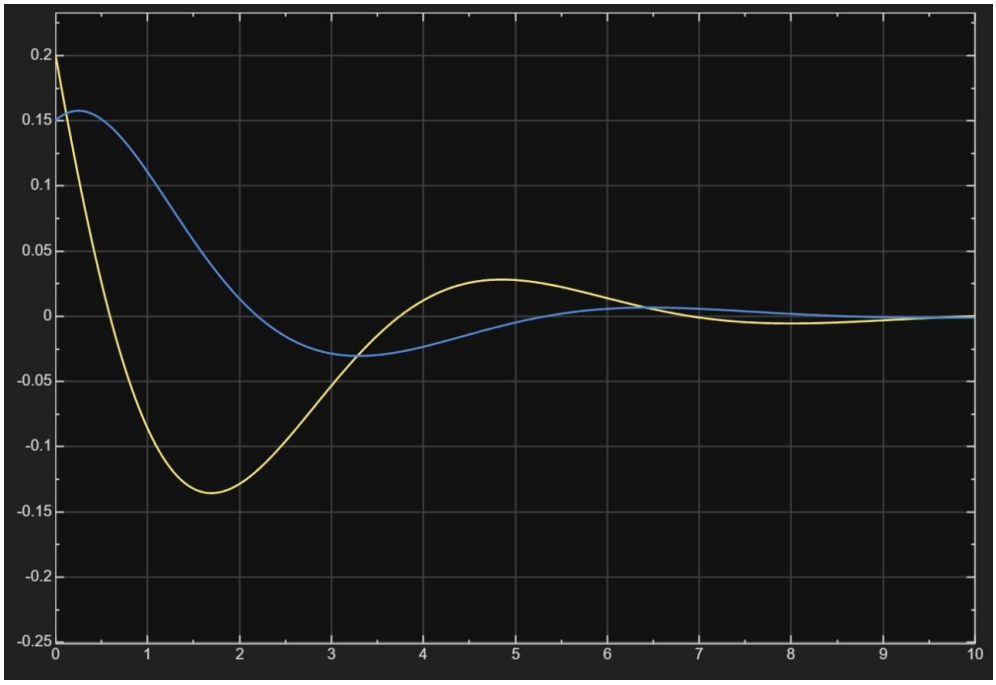


Kp1 = 400, Ki1 = 40
Kp2 = 400, Ki2 = 40

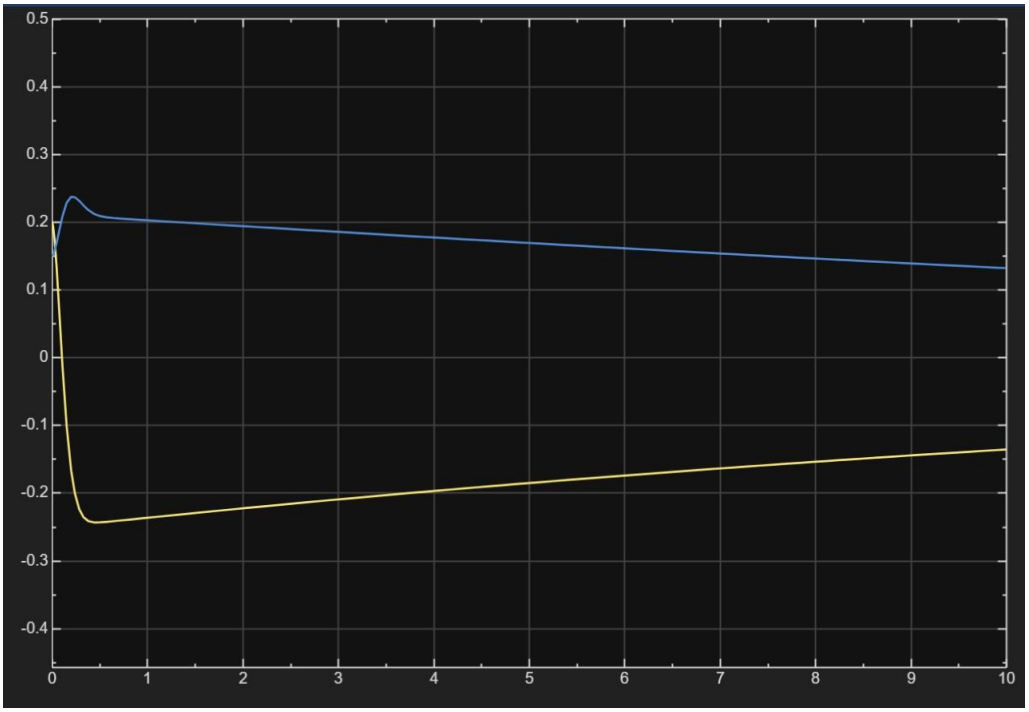


4.4 PID CONTROLLER

Kp1 = 200, Ki1 = 200, Kd1 = 250
Kp1 = 200, Ki1 = 200, Kd2 = 250

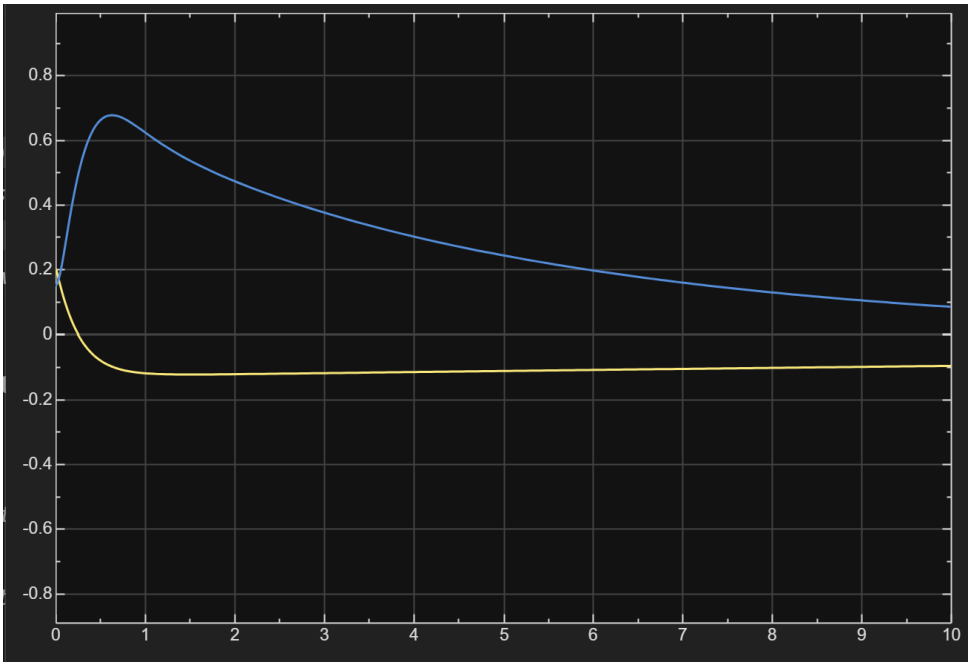


Kp1 = 150, Ki1 = 20, Kd1 = 10
Kp1 = 150, Ki1 = 20, Kd2 = 10



Kp1 = 300, Ki1 = 10, Kd1 = 80

Kp1 = 45, Ki1 = 12, Kd2 = 15



5 Conclusion

5.1 Proportional Derivative Controller

5.1.1 Effect of Derivative Gain (Kd)

Elevated KD Values:

Benefits:

- Enhanced derivative gain substantially suppresses oscillatory motion and curtails the magnitude of repetitive fluctuations, producing steadier and more refined control behavior.
- The robotic manipulator exhibits greater resistance to unwanted vibrations, a crucial factor when executing tasks demanding high precision and smooth motion trajectories.

Drawbacks:

- Excessively high derivative gain parameters can push the system into an over-damped state, compromising response speed and degrading dynamic performance.
- This excessive damping effect may diminish the system's ability to react promptly, creating challenges in applications requiring swift corrective actions or fast-paced maneuvering.

Reduced KD Values:

Benefits:

- Lower derivative gain introduces minimal damping force, enabling the system to execute faster transitional movements when responding to input changes. This characteristic proves valuable when prioritizing response speed over motion smoothness.
- The manipulator demonstrates increased nimbleness, making it well-suited for operations that demand quick and accurate positional adjustments.

Drawbacks:

- Inadequate damping associated with reduced KD settings may trigger pronounced oscillatory behavior, leading to significant overshoot and compromised stability that undermines positioning accuracy and control precision.

a) Controlling Overshoot

High KD Values:

Benefits:

- Higher derivative gain helps reduce overshoot and stops excessive oscillations, making the movement smoother and more controlled. This is important when you need precise and stable positioning.

Drawbacks:

- If the derivative gain is too high, the system becomes sluggish and takes longer to settle at the target position. This delay can be a problem when quick movements are needed.

Low KD Values:

Benefits:

- Lower derivative gain allows the system to respond faster initially, even if it means some overshoot occurs. This can be helpful when speed matters more than perfect positioning.

Drawbacks:

- Very low derivative gain may not reduce overshoot enough, allowing the arm to keep oscillating back and forth. This reduces accuracy, which is a problem when exact positioning is required.

C) Impact on Response Speed

High KD Values:

Benefits:

- When properly tuned, higher derivative gain can shorten the stabilization period, making the system reach its target faster and more efficiently for accurate positioning tasks.

Drawbacks:

- In certain situations, setting the derivative gain too high may either create unwanted overshoot or slow down how quickly the system settles, which becomes problematic during fast-paced operations.

Low KD Values:

Benefits:

- Lower derivative gain keeps the initial response quick, even though it might take a bit longer for the system to completely settle down. This works well when you need immediate action rather than waiting for perfect stability.

Drawbacks:

- Very low derivative gain can cause the system to keep oscillating for too long, extending the time before it settles, which creates problems when you need both accuracy and quick stabilization.

High KD Values:

Benefits:

- Higher derivative gain helps maintain system stability when configured correctly, particularly in applications requiring smooth, controlled, and accurate movements without unwanted oscillations.

Drawbacks:

- Excessive derivative gain can make the system over-damped, causing it to settle slowly and limiting its ability to handle rapid changes effectively. In some cases, this might even lead to stability issues.

Low KD Values:

Benefits:

- Lower derivative gain keeps the system more responsive by avoiding over-damping, making it better for dynamic tasks where quick reactions and flexibility are important.

Drawbacks:

- Too little derivative gain may not control oscillations properly, making the system unstable with excessive overshoot or unpredictable behavior that can compromise overall stability.

5.1.2 Conclusion

Fine-tuning the derivative gain (KD) in a PD control system requires finding the right balance among reducing overshoot, managing oscillations, and preserving system responsiveness. The optimal KD value depends on the system's mechanical characteristics, performance requirements, and the specific operations being executed. Proper adjustment is crucial to ensure the robotic arm functions effectively, delivering the appropriate balance among speed, accuracy, and stability.

5.2 Proportional Integral Controller

5.2.1 Effect of Proportional Gain (Kp)

High Kp Values:

Benefits:

- Raising the proportional gain makes the controller react more strongly to joint angle errors, providing more forceful corrections when the system moves away from its target position.
- Higher proportional gain enables faster approach to the desired joint angles, even when dealing with large position errors.

Drawbacks:

- Setting Kp too high can cause the system to overshoot its target, possibly creating oscillations or unstable behavior that makes precise control difficult.
- High proportional gain may make the system too sensitive to noise and external disturbances, resulting in jerky or inconsistent movements.

Low Kp Values:

Benefits:

- Lower proportional gain produces a gentler and steadier response to errors, which works well when maintaining stability is more important than speed.
- It minimizes overshoot and oscillations, resulting in smoother and more predictable system behavior.

Drawbacks:

- Very low proportional gain can make the system respond too slowly, taking a long time to reach the target position, particularly when starting with large errors.
- It may cause delays in achieving the desired joint angles, which becomes an issue in applications where time matters.

5.2.2 Effect of Integral Gain (Ki)

High Ki Values:

Benefits:

- Raising the integral gain strengthens the controller's capability to remove steady-state errors by accumulating error over time, ensuring the system reaches and stays at the desired joint positions.
- Higher integral gain enhances positioning accuracy, helping the system settle closer to the target with minimal long-term drift.

Drawbacks:

- Too much integral gain can create instability, particularly in noisy conditions or when disturbances are present. This may cause overshoot, oscillations, or unpredictable behavior.
- High integral gain demands precise tuning and may not work well for all system types or operating conditions.

Low Ki Values:

Benefits:

- Lower integral gain offers more stable operation with less sensitivity to noise or external disturbances, reducing the chance of oscillations or overshoot.
- Reduced integral gain works well when eliminating steady-state error completely isn't essential.

Drawbacks:

- Very low integral gain may allow steady-state errors to remain, limiting the system's ability to hold accurate joint positions over extended periods, especially in precision-critical applications.

i) Correcting Steady-State Error

High Ki Values:

Benefits:

- High integral gain efficiently eliminates steady-state errors, helping the system converge accurately to target joint angles without lasting deviations.

Drawbacks:

- Excessive integral gain can make the system unstable, raising the probability of overshoot and oscillations, especially in systems with considerable noise or disturbances.

Low Ki Values:

Benefits:

- Lower integral gain produces a more stable system, though it may permit small steady-state errors to persist.

Drawbacks:

- Very low integral gain might not adequately eliminate steady-state errors, potentially reducing long-term system accuracy.

5.2.3 Conclusion

Tuning a PI controller involves finding the right balance between Kp and Ki to achieve good performance while preventing instability, overshoot, or oscillations. The best gain values depend on the robotic arm's specific characteristics and what the application needs from the control system.

5.3 Proportional Integral Derivative Controller

5.3.1 Effect of Proportional Gain (Kp)

High Kp Values:

Benefits:

- Minimized steady-state error: Higher proportional gain rapidly decreases steady-state error by generating control actions proportional to the current error magnitude. This creates a stronger response when errors are large, enabling faster movement toward the target position.
- Better accuracy: Elevated Kp values improve precision and positioning performance, making them ideal for applications demanding fine control, such as robotic manipulators.

Drawbacks:

- Oscillation risk: Overly high proportional gain can trigger oscillatory motion or unstable behavior, making it difficult to achieve smooth and accurate control.
- Overshoot and unpredictability: Excessive Kp may cause the system to overshoot its target before stabilizing, potentially creating unstable or inconsistent responses.

Low Kp Values:

Benefits:

- Better stability: Reduced proportional gain decreases the chance of oscillations and instability, producing a more controlled and steady error response.

Drawbacks:

- Slower correction: Low proportional gain creates a less forceful control action, which slows error correction. Consequently, the system takes more time to arrive at the desired position.

5.3.2 Effect of Derivative Gain (Kd)

High Kd Values:

Benefits:

- Oscillation dampening: Higher derivative gain effectively slows the rate at which error changes, reducing oscillations and creating smoother system behavior.
- Enhanced stability: Properly adjusted Kd provides greater stability, making it perfect for applications needing high precision and steady control, such as robotic systems.

Drawbacks:

- Over-damping: Excessive derivative gain can create an over-damped response where the system becomes slow and takes too long to settle. This may limit the system's ability to handle dynamic changes, reducing its responsiveness.

Low Kd Values:

Benefits:

- Quicker initial response: Lower derivative gain enables the system to react swiftly to errors and move rapidly toward the target position.

Drawbacks:

- Insufficient damping: Very low derivative gain may not adequately reduce oscillations, potentially compromising system stability.

5.3.3 Effect of Integral Gain (Ki)

High Ki Values:

Benefits:

- Eliminates persistent errors: Higher integral gain effectively removes long-term, ongoing errors by accumulating historical errors and modifying the control output accordingly. This ensures the system reaches its target despite steady-state disturbances.

Drawbacks:

- Instability risk: Excessive integral gain can cause instability, particularly in systems affected by noise or disturbances. This may produce unpredictable and erratic control behavior.

Low Ki Values:

Benefits:

- Maintains stability: Lower integral gain prevents excessive error accumulation, reducing oscillation risk and preserving overall system stability. This creates smoother and more gradual control.

Drawbacks:

- Persistent steady-state error: Inadequate integral gain may fail to remove steady-state errors, leaving lasting inaccuracies in system performance.

Conclusion:

Comparison of PD, PI and PID Controllers:

Controller	Action / Effect	Steady-state error	Response speed	Overshoot	Stability
PD (Proportional-Derivative)	Proportional reduces error; Derivative predicts error trend and damps motion (rate feedback).	Does not eliminate steady-state error (unless P large or there is plant type).	Fast (derivative improves transient), reduces rise time.	Reduces overshoot compared to pure P if tuned well.	Improves damping; can increase closed-loop stability margin when tuned properly.
PI (Proportional-Integral)	Proportional reduces present error; Integral accumulates past error and removes steady-state offset.	Eliminates steady-state error for step inputs (integral action).	Slower than PD for transients (integrator can slow response).	Can increase overshoot and oscillation if integral action too strong.	Integrator can reduce phase margin; may make system less stable if aggressive.
PID (Proportional-Integral-Derivative)	Combines immediate correction, elimination of steady-state error, and predictive damping for transients.	Can drive steady-state error to zero (with I).	Can be tuned for fast response while controlling overshoot (best transient shaping).	Can achieve low or moderate overshoot if D tuned correctly; risk if I too large.	Most flexible – can achieve good stability and transient performance when tuned correctly.

Performance Comparison:

Metric	PD	PI	PID
Steady-state error (step)	Usually nonzero (no integral).	Zero for step inputs (integrator removes offset).	Can be zero for step inputs (with proper tuning).
Transient speed (rise time)	Fast (D improves speed/damping).	Slower than PD; integrator can slow initial response.	Can be fast (combines P and D benefits).
Overshoot	Lower than P alone if D applied, but may still overshoot depending on Kp.	Can increase overshoot if integral aggressive.	Can be controlled (D reduces overshoot), depends on tuning.
Settling time	Typically, shorter than P or PI for same steady-state error.	Potentially longer (oscillations from I).	Can be short if tuned well; D helps damping.
Robustness to model error	Good if D and P tuned; not ideal if plant uncertain.	Moderate – integral helps steady behavior but can destabilize if plant varies.	Best of the three when tuned correctly (most flexible).

Tuning a PID controller requires finding the right balance among Kp, Ki, and Kd to achieve optimal control performance. The objective is to minimize steady-state error, prevent overshoot or oscillations, and ensure system stability. The best tuning parameters depend on the robotic arm's specific behavior and what the application requires.