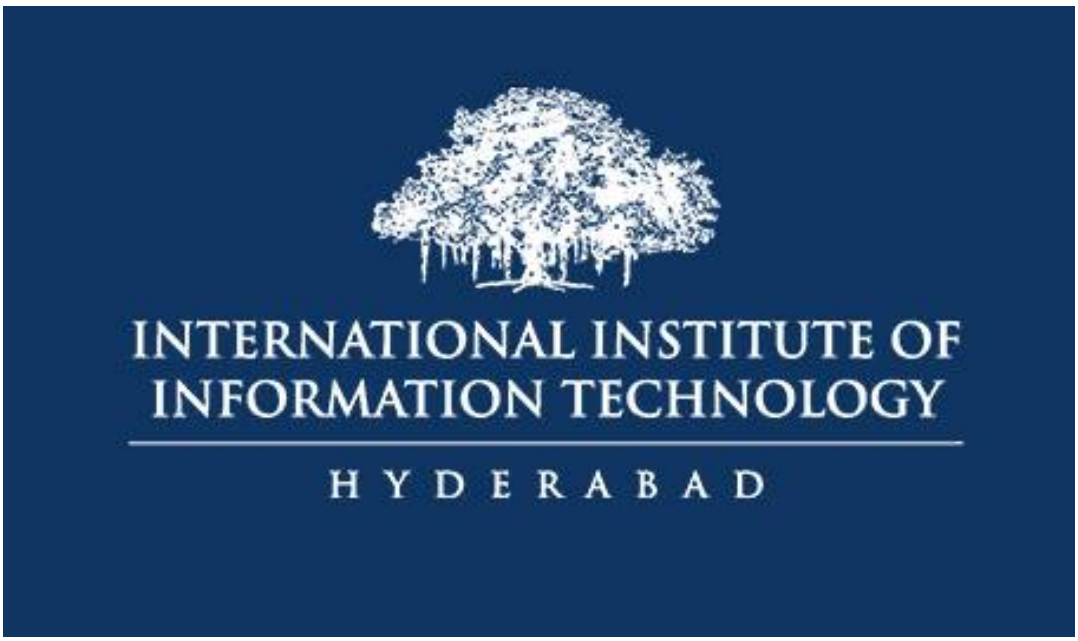
Systems Thinking Project

Controlling 2 link Serial Planar Manipulator

Team Name: LUSR



Jay Daulatkar
2023102044

Parth Tokekar
2023102041

Jainil Bavishi
2023102045

Vansh Agarwal
2023102043

Dhimant Bhuva
2023102063

Jal Parikh
2023102066

Vaibhav Wadhwani
2023102058

Devansh Varshney
2023102042

Abstract

This report investigates the use of state functions and variables to model the dynamics of a 2-link robotic manipulator. The objective is to transition the system from its initial state to its final state while optimizing control performance. In this study, PI, PD, and PID controllers are implemented to regulate the manipulator's complex, non-linear behavior. The focus is on achieving accurate position control through torque modulation. We derive the equations of motion and conduct simulations using MATLAB-Simulink to validate the proposed approach.

Contents

1 Introduction

The system in question is a 2-link robotic arm, which can be modeled similarly to a double pendulum. It consists of two connected links, with a movable joint between them allowing relative motion, and a hinged joint that secures the first link to a fixed base (the floor). The task at hand, as outlined in the problem, is to design control mechanisms that steer the system from its initial configuration of [0.1 rad, 0.1 rad], where both joints are slightly displaced, to the desired final configuration of [0 rad, 0 rad], where both joints are aligned at rest.

To achieve this, we propose the use of a proportional-integral-derivative (PID) controller, which is known for its effectiveness in managing dynamic, nonlinear systems like robotic manipulators. The controller applies a combination of proportional, integral, and derivative actions to generate corrective signals based on the error between the current and desired states. This allows the system to efficiently converge to the target configuration.

The controller is designed to not only transition the system to the [0 rad, 0 rad] state but also to ensure that the process is smooth and that any steady-state error— the difference between the desired position and the actual final position — is minimized. To achieve this, the PID controller modulates two independent torque inputs, one for each joint of the robotic arm, thereby controlling the motion and position of the links in real time.

Through this approach, we aim to ensure that the system reaches its final state with minimal oscillations, rapid settling time, and precise positioning, while maintaining stability throughout the process. The application of two separate torques enables fine-tuned control of each link, ensuring that the desired result is obtained efficiently and accurately.

1.1 Conceptual Modeling

The robotic arm can be modeled as two mass-less rods, each with a mass at its tip. In this setup, one rod and its corresponding mass represent the arm, while the other rod and mass represent the forearm. Additional masses are positioned at the elbow joint and the end of the forearm. By connecting these two systems, we can study the overall dynamics of the arm. Both systems are free to oscillate within a plane.

1.2 Theoretical Framework

The dynamics of the manipulator can be fully characterized by defining its Lagrangian, which depends on both its potential and kinetic energies. By applying the Euler-Lagrange equation, we can determine the torques exerted on each link. Then, PI, PD, or PID controllers are used to simulate how the system behaves under different torque inputs.

1.3 Methodology

The motion equation for this 2-link manipulator is a non-linear differential equation. We solve it using numerical methods and utilize MATLAB for simulating and visualizing the control strategies.

1.4 Objective

The objectives are to model the 2-link robotic manipulator using state functions, apply and compare PI, PD, and PID controllers for system control, and simulate the manipulator's dynamics using MATLAB.

The structure of this paper is as follows: Section 2 covers the mathematical modeling, Section 3 outlines the design of the controllers, Section 4 presents the simulation results, and Section 5 concludes the study with recommendations for future work.

2 Derivations of 2 Link Manipulator Equations

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

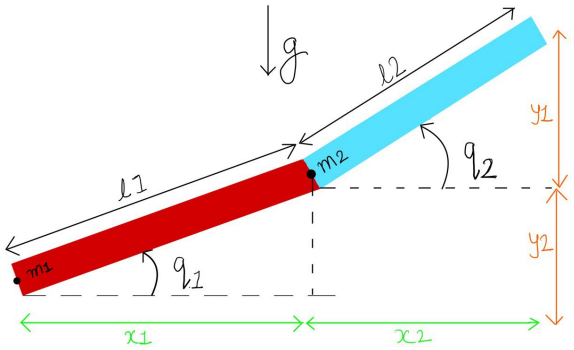


Figure 1: Two Link Manipulator

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

$$x_1 = l_1 \cos(q_1)$$
$$y_1 = l_1 \sin(q_1)$$
$$x_2 = l_1 \cos(q_1) + l_2 \cos(q_2)$$
$$y_2 = l_1 \sin(q_1) + l_2 \sin(q_2)$$

Therefore, the velocities of the two masses will be:

$$v_1 = \sqrt{(\dot{x}_1)^2 + (\dot{y}_1)^2}$$

$$v_2 = \sqrt{(\dot{x}_2)^2 + (\dot{y}_2)^2}$$

where,

$$\dot{x}_1 = -l_1 \dot{q}_1 \sin q_1$$
$$\dot{y}_1 = l_1 \dot{q}_1 \cos q_1$$

$$\dot{x}_2 = -l_1\dot{q}_1 \sin q_1 - l_2\dot{q}_2 \sin q_2$$

$$\dot{y}_2 = l_1\dot{q}_1 \cos q_1 + l_2\dot{q}_2 \cos q_2$$

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

Now, the total kinetic energy can be given by:

$$KE = \frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2}$$

$$KE = \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2)$$

$$KE = \frac{1}{2}m_1\left((-l_1\dot{q}_1 \sin q_1)^2 + (l_1\dot{q}_1 \cos q_1)^2\right) + \frac{1}{2}m_2\left((-l_1\dot{q}_1 \sin q_1 - l_2\dot{q}_2 \sin q_2)^2 + (l_1\dot{q}_1 \cos q_1 + l_2\dot{q}_2 \cos q_2)^2\right.$$

$$KE = \frac{1}{2}(m_1 + m_2)l_1^2\dot{q}_1^2 + \frac{1}{2}(m_2)l_2^2\dot{q}_2^2 + \frac{1}{2}m_2 l_1 l_2 \dot{q}_1\dot{q}_2 \cos(q_1 - q_2)$$

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

$$PE = \sum_{i=1}^{2} PE_i(q) = \sum_{i=1}^{2} m_i g h_i(q)$$

$$PE = (m_1 + m_2)gl_1 \sin q_1 + m_2 g l_2 \sin q_2$$

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

Next by Lagrange Dynamics, the Lagrangian L is defined as:

$$L = KE - PE$$

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

$$L = \frac{1}{2}(m_1+m_2)l_1^2\dot{q}_1^2 + \frac{1}{2}(m_2)l_2^2\dot{q}_2^2 + \frac{1}{2}m_2 l_1 l_2 \dot{q}_1\dot{q}_2 \cos(q_1-q_2) - (m_1+m_2)gl_1 \sin q_1 - m_2 g l_2 \sin q_2$$

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

$$l_1\ddot{q}_1 + \delta l_1 \dot{q}_1^2 \cos q_1 - q_2 = \frac{\delta T_1}{m_2 l_1} - \delta l_2 \dot{q}_2^2 \sin(q_1 - q_2) - g \cos q_1$$

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

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

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

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

where,

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

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

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

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

After differentiating with respect to time,

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

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

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

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

$$\begin{aligned}
s_1 = u_3, \\
s_2 = u_4, \\
s_3 = g_1(t, u_1, u_2, u_3, u_4), \\
s_4 = g_2(t, u_1, u_2, u_3, u_4)
\end{aligned}$$

The initial conditions are given by

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

Where,

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

3 Dynamics And Modelling

3.1 Dynamics of a Two Link Manipulator

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

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

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

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

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

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

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

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

4 Circuit Block Diagram

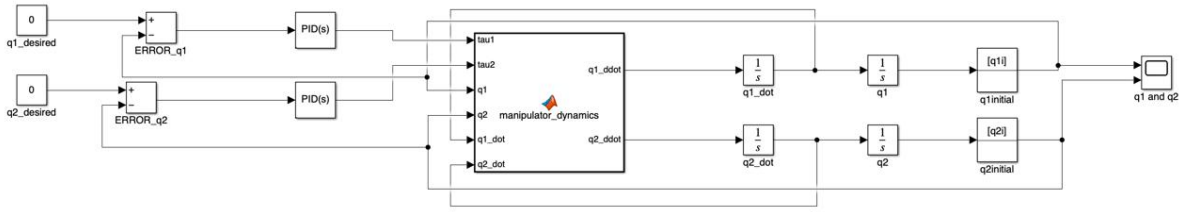


Figure 2: Block Diagram of Two Link Manipulator

The block diagram represents a control system for a two-link manipulator. Below is a detailed explanation of each component and the flow of signals:

1. **Desired Angles Input ($q1_{\text{desired}}$ and $q2_{\text{desired}}$):**
   - These are the desired angles for the joints of the manipulator.
2. **Error Calculation (ERROR_q1 and ERROR_q2):**
   - The error signals are calculated by subtracting the actual angles ($q1$ and $q2$) from the desired angles ($q1_{\text{desired}}$ and $q2_{\text{desired}}$).
3. **PID Controllers:**

- Two PID controllers are used to process the error signals and generate the control torques ($\tau 1$ and $\tau 2$).
- The output of the PID controllers is fed into the manipulator dynamics block.

4. **Manipulator Dynamics:**
   - This block represents the dynamics of the two-link manipulator.
   - It takes the control torques ($\tau 1$ and $\tau 2$), current joint angles ($q1$ and $q2$), and their derivatives ($\dot{q1}$ and $\dot{q2}$) as inputs.
   - The outputs are the accelerations of the joint angles ($\ddot{q1}$ and $\ddot{q2}$).
5. **Integration Blocks:**
   - There are two sets of integrators. The first set integrates the accelerations ($\ddot{q1}$ and $\ddot{q2}$) to get the velocities ($\dot{q1}$ and $\dot{q2}$).
   - The second set integrates the velocities to get the angles ($q1$ and $q2$).
6. **Initial Conditions:**
   - Initial conditions for the integrators are provided by blocks labeled as $[q1i]$ and $[q2i]$.
7. **Output:**
   - The final output consists of the joint angles $q1$ and $q2$, which are displayed or used for further processing.

5 Understanding Control Systems For Implementation

Simulations of the 2-link robotic manipulator are implemented in MATLAB. The joint angles are controlled through the designed controllers. The results of different control strategies are analyzed.

5.1 P Controller

5.1.1 Introduction to P Control

A **Proportional (P) controller** is a fundamental control strategy used in various engineering applications. The primary objective of a P controller is to minimize the error between the desired setpoint and the actual output of a system by applying a corrective action that is proportional to the error. The control law for a P controller is expressed as:

$$f(t) = K_P e(t)$$

Where:

- $\mathbf{e(t)}$: is the error at time t between the desired setpoint and the current state.
- $\mathbf{K_P}$: is the proportional gain, which determines the magnitude of the correction applied based on the current error.

The proportional term provides an immediate response to the current error, driving the system towards the desired setpoint.

5.1.2 Application to a Double Joint System

In a double-joint robotic system, the joints are controlled independently, each with its own P controller. The control forces applied to the joints are f_1 and f_2, and the control equations for each joint can be formulated as:

$$f_1 = K_{P1}(q_{1f} - q_1)$$
$$f_2 = K_{P2}(q_{2f} - q_2)$$

Where:

- $\mathbf{q_1}$ and $\mathbf{q_2}$: the actual positions of the joints.
- $\mathbf{q_{1f}}$ and $\mathbf{q_{2f}}$: the desired (reference) positions for the joints.
- $\mathbf{K_{P1}}, \mathbf{K_{P2}}$: the proportional gains for joint 1 and joint 2, respectively.

The P controllers ensure that both joints minimize the position error, moving toward their respective desired positions. The proportional component drives the joints towards the target positions based on the current error.

5.1.3 Dynamic Model of the System

The dynamics of the double-joint system are described by a second-order differential equation that captures the effects of inertia, Coriolis forces, and gravitational forces:

$$\ddot{q} = -M^{-1}(q)[C(q, \dot{q}) + G(q)] + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Where:

- $\mathbf{M}(\mathbf{q})$: the mass (inertia) matrix, which varies with the configuration of the system.
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$: represents Coriolis and centripetal forces arising from joint motion.
- $\mathbf{G}(\mathbf{q})$: represents the gravitational forces acting on the joints.

The control signals f_1 and f_2 are calculated from the P control laws and are applied to each joint to drive the system towards the desired configuration. The torque generated at each joint is given by:

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

Substituting the control equations for f_1 and f_2 into the torque equations provides the final expressions for the joint torques:

$$\tau_1 = M_1 L_1^2 K_{P1}(q_{1f} - u_1) + M_2 L_1 L_2 \cos(u_1 - u_2) K_{P2}(q_{2f} - u_2)$$

$$\tau_2 = M_2 L_1 L_2 \cos(u_1 - u_2) K_{P1}(q_{1f} - u_1) + M_2 L_2^2 K_{P2}(q_{2f} - u_2)$$

These equations describe the torques required to drive the system based on the current and desired positions of the joints.

5.1.4 Behavior of the P Controller

The P controller design is represented by the control law:

$$K_P$$

This control law indicates that the system response is directly proportional to the current error.

Proportional Action: The proportional term provides immediate corrective action based on the current error, driving the system towards the desired state.

Compared to a Proportional-Integral (PI) controller, which reduces steady-state error through integration, the P controller offers faster responses but may result in steady-state error if the proportional gain is not sufficiently high.

5.1.5 Simulation

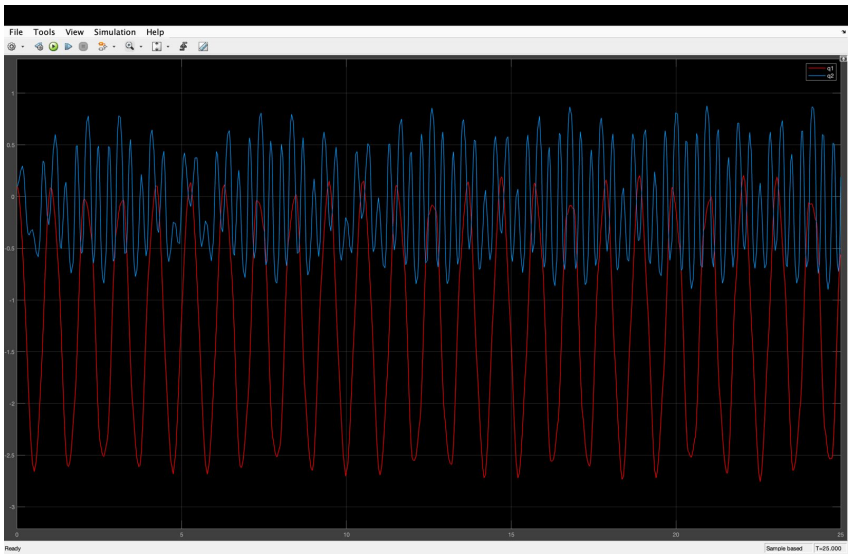


Figure 3: $K_p = 5$

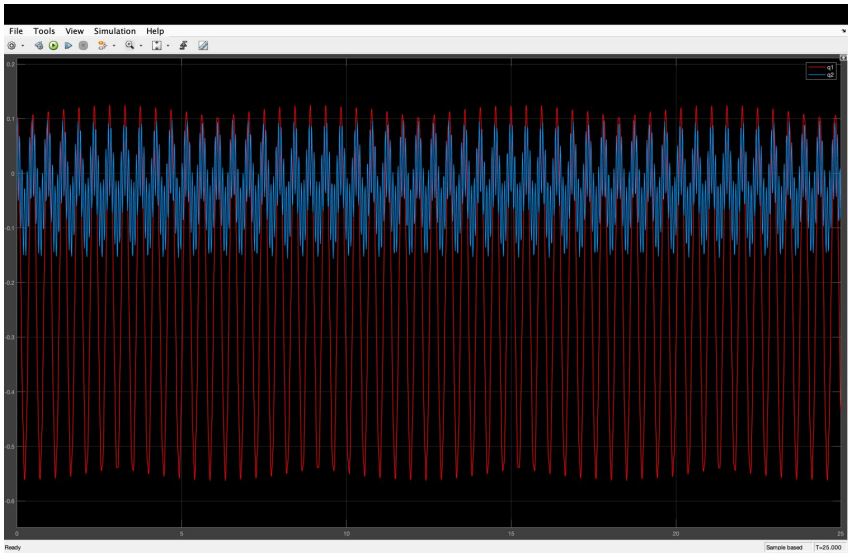


Figure 4: $K_p = 150$

5.2 PI Controller

5.2.1 Introduction to PI Control

A **Proportional-Integral (PI) controller** is a widely used control strategy in many engineering systems. It is designed to minimize the difference, or error, between the desired state and the actual state of a system. This is accomplished by using two terms: a proportional term and an integral term. The control law for a PI controller is expressed as:

$$f(t) = K_P e(t) + K_I \int e(t)dt$$

Where:

- $\mathbf{e(t)}$: is the error at time t between the desired setpoint and the current state.
- $\mathbf{K_P}$: is the proportional gain, responsible for correcting the error based on its current magnitude.
- $\mathbf{K_I}$: is the integral gain, which accumulates past errors over time and ensures that the system eliminates steady-state error.

The proportional term provides a response proportional to the current error, while the integral term corrects long-term discrepancies by integrating the error over time. Together, these components help the system reach the desired setpoint with minimal error, both in the short and long term.

5.2.2 Application to a Double Joint System

We extend the concept of PI control to a double-joint robotic system. In this system, the joints are controlled independently, and each joint is equipped with its own PI controller. The inputs to the system are f_1 and f_2, which are the control forces applied to the joints. The control equations for each joint are formulated as:

$$f_1 = K_{P1}(q_{1f} - q_1) + K_{I1} \int (q_{1f} - q_1)dt$$

$$f_2 = K_{P2}(q_{2f} - q_2) + K_{I2} \int (q_{2f} - q_2)dt$$

- $\mathbf{q_1}$ and $\mathbf{q_2}$: the actual positions of the joints.
- $\mathbf{q_{1f}}$ and $\mathbf{q_{2f}}$: the desired (reference) positions for the joints.
- $\mathbf{K_{P1}}, \mathbf{K_{I1}}$: the proportional and integral gains for joint 1.
- $\mathbf{K_{P2}}, \mathbf{K_{I2}}$: the proportional and integral gains for joint 2.

The PI controllers ensure that both joints minimize the position error, moving toward their respective desired positions. The proportional component drives the joints towards the target positions, while the integral component helps eliminate any steady-state error by considering the history of the error.

5.2.3 Dynamic Model of the System

The motion of the double-joint system is governed by complex dynamics, which include mass, inertia, Coriolis effects, and gravitational forces. The system dynamics are represented by the following second-order differential equation:

$$\ddot{q} = -M^{-1}(q)[C(q, \dot{q}) + G(q)] + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Where:

- $\mathbf{M}(\mathbf{q})$: the mass (inertia) matrix, which varies with the configuration of the system.
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$: represents Coriolis and centripetal forces that arise due to the motion of the joints.
- $\mathbf{G}(\mathbf{q})$: represents the gravitational forces acting on the joints.

The control signals f_1 and f_2 are calculated from the PI control laws and applied to each joint to drive the system towards the desired configuration. The torque generated at each joint is given by:

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

Substituting the control equations for f_1 and f_2 into the torque equations gives the final expressions for the joint torques:

$$\tau_1 = M_1 L_1^2 (K_{P1}(q_{1f} - u_1) + K_{I1} u_3) + M_2 L_1 L_2 \cos(u_1 - u_2)(K_{P2}(q_{2f} - u_2) + K_{I2} u_4)$$

$$\tau_2 = M_2 L_1 L_2 \cos(u_1 - u_2)(K_{P1}(q_{1f} - u_1) + K_{I1} u_3) + M_2 L_2^2 (K_{P2}(q_{2f} - u_2) + K_{I2} u_4)$$

These equations represent the torques required to drive the system based on the current and desired positions of the joints.

5.2.4 Behavior of the PI Controller

The PI controller design is represented by the control law:

$$K_P + \frac{K_I}{s}$$

This introduces a third root into the system, effectively making the system third-order. The proportional term K_P governs the rate at which the system responds to current errors, while the integral term K_I ensures that the system compensates for any accumulated steady-state error.

Proportional Action: The proportional term provides immediate corrective action based on the current error, driving the system towards the desired state.

Integral Action: The integral term accumulates past errors, allowing the system to correct any persistent deviations from the desired state.

Compared to a Proportional-Derivative (PD) controller, which tends to provide faster responses with reduced oscillations, the PI controller sacrifices some transient speed for improved steady-state accuracy. The PD controller is generally more effective at reducing oscillations and improving settling time, while the PI controller ensures that the error goes to zero over time.

5.2.5 Simulation

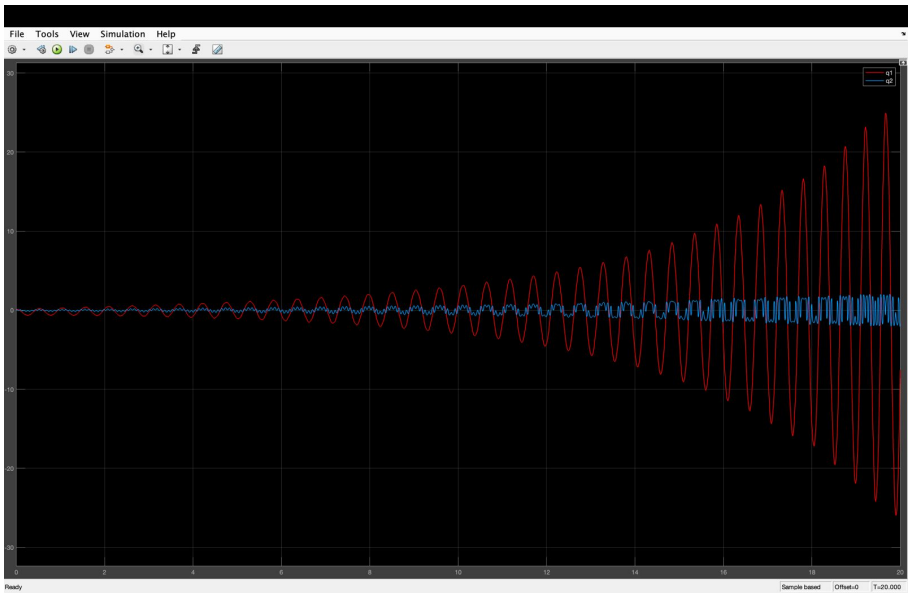


Figure 5: $K_p = 120$ and $K_I = 50$

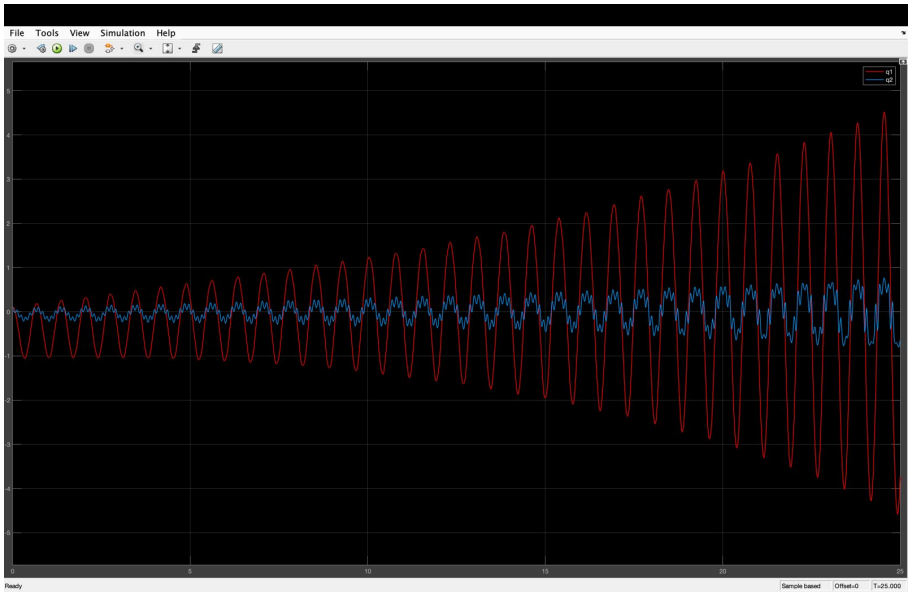


Figure 6: $K_p = 600$ and $K_I = 10$

5.3 PD Controller

5.3.1 Introduction to PD Control

A **Proportional-Derivative (PD) controller** is a widely employed control mechanism in dynamic systems, known for its ability to minimize oscillations and improve transient responses. The control law for a PD controller combines two terms: a proportional term and a derivative term, both of which influence the control signal based on the system's current state and rate of change. The control equation is expressed as:

$$f(t) = K_P e(t) + K_D \frac{d}{dt} e(t)$$

Where:

- $\mathbf{e(t)}$: is the error between the desired setpoint and the actual state at time t.
- $\mathbf{K_P}$: is the proportional gain, which responds to the magnitude of the current error.
- $\mathbf{K_D}$: is the derivative gain, which considers the rate of change of the error and works to dampen oscillations.

The PD controller effectively balances the system's response by providing immediate corrective action for the current error while also damping the system's motion through its derivative term. This helps reduce overshooting and oscillations, making PD control well-suited for systems requiring faster stabilization.

5.3.2 Application to a Double Joint System

In the context of a double-joint system, PD control can be applied to control the motion of each joint independently. The system uses two inputs, f_1 and f_2, to control the positions of two joints. The control equations for the two joints are given by:

$$f_1 = K_{P1}(q_{1f} - q_1) - K_{D1}\dot{q}_1$$

$$f_2 = K_{P2}(q_{2f} - q_2) - K_{D2}\dot{q}_2$$

Where:

- $\mathbf{q_1}$ and $\mathbf{q_2}$: are the current positions of joints 1 and 2, respectively.
- $\mathbf{q_{1f}}$ and $\mathbf{q_{2f}}$: are the desired positions for joints 1 and 2, respectively.
- $\mathbf{K_{P1}}$ and $\mathbf{K_{D1}}$: are the proportional and derivative gains for joint 1, respectively.
- $\mathbf{K_{P2}}$ and $\mathbf{K_{D2}}$: are the proportional and derivative gains for joint 2, respectively.

The proportional terms $K_{P1}(q_{1f} - q_1)$ and $K_{P2}(q_{2f} - q_2)$ drive the joints towards their desired positions, while the derivative terms $-K_{D1}\dot{q}1$ and $-KD2\dot{q}_2$ act to reduce oscillations by damping the system's velocity.

5.3.3 Dynamic Model of the System

The dynamic behavior of the double-joint system is influenced by factors such as mass, inertia, Coriolis forces, and gravitational effects. The system dynamics are represented by the following second-order differential equation:

$$\ddot{q} = -M^{-1}(q)[C(q, \dot{q}) + G(q)] + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Where:

- $\mathbf{M(q)}$: is the mass matrix, which depends on the system's configuration.
- $\mathbf{C(q, \dot{q})}$: represents Coriolis and centrifugal forces.
- $\mathbf{G(q)}$: accounts for gravitational forces acting on the joints.

The control signals f_1 and f_2, calculated using the PD controller, are applied to the system to regulate joint movement. The torques at the joints are calculated by:

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

Substituting the expressions for f_1 and f_2 gives the torque equations:

$$\tau_1 = (M_1{+}M_2)L_1^2 \left(K_{P1}(q_{1f} - u_1) - K_{D1}u_3\right){+}M_2L_1L_2\cos(u_1{-}u_2)\left(K_{P2}(q_{2f} - u_2) - K_{D2}u_4\right)$$

$$\tau_2 = M_2L_1L_2\cos(u_1 - u_2)\left(K_{P1}(q_{1f} - u_1) - K_{D1}u_3\right) + M_2L_2^2\left(K_{P2}(q_{2f} - u_2) - K_{D2}u_4\right)$$

These equations describe the torques required to control each joint, considering the proportional and derivative control laws.

5.3.4 Behavior of the PD Controller

The PD controller is defined by the control law:

$$K_P + K_D s$$

Where s is the Laplace variable, representing the system's derivative term. The PD controller introduces a derivative block in parallel with the proportional gain, contributing to the system's stability by damping oscillations and improving the transient response.

Proportional Action: The proportional term K_P responds to the current error, driving the system toward the desired setpoint. This term is responsible for minimizing the magnitude of the error.

Derivative Action: The derivative term K_D responds to the rate of change of the error, acting as a damper that reduces oscillations and prevents overshooting. The derivative action helps smooth out the transient response by counteracting rapid changes in error.

The PD controller's effectiveness in reducing oscillations is particularly important in systems with underdamped responses, where the system may exhibit oscillatory behavior. By tuning the derivative gain K_D, we can mitigate oscillations and improve settling time.

However, while the derivative term helps stabilize the transient response, it does not eliminate steady-state error, which is a limitation of PD control compared to PI control. Therefore, in scenarios where eliminating steady-state error is critical, a PI or PID controller might be more suitable.

5.3.5 Simulation

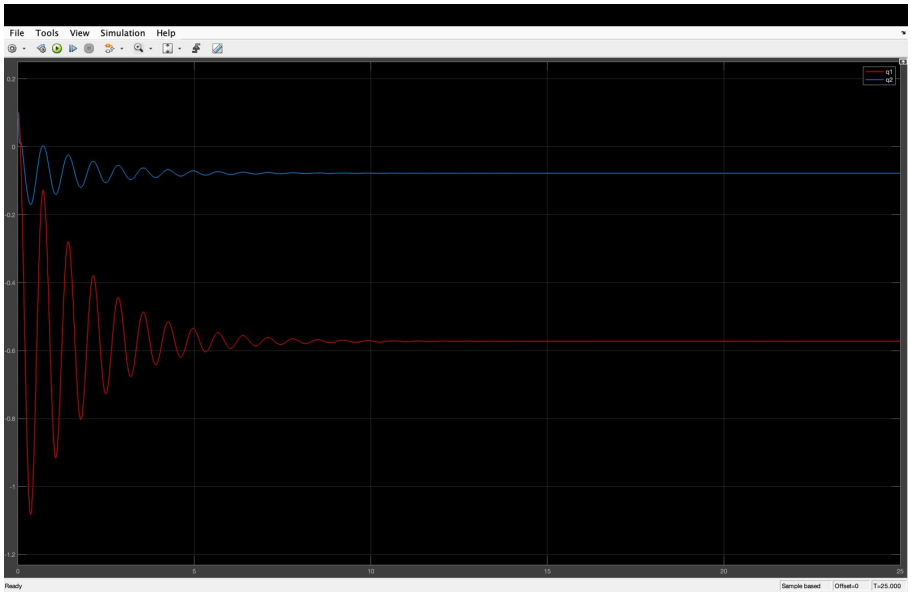


Figure 7: $K_p = 50$ and $K_D = 1$

Figure 8: $K_p = 50$ and $K_D = 50$

5.4 PID Controller

5.4.1 Introduction to PID Control

The PID (Proportional-Integral-Derivative) controller is a fundamental and versatile control strategy used in dynamic systems to achieve desired performance. The controller combines three distinct control actions—proportional, integral, and derivative—each contributing uniquely to the overall response of the system. The control law for a PID controller is expressed as:

$$f(t) = K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$$

Where:

- $\mathbf{e(t)}$: is the error between the desired setpoint and the current state.
- $\mathbf{K_P}$: is the proportional gain, which reacts to the present error.
- $\mathbf{K_I}$: is the integral gain, which accumulates past errors and eliminates steady-state error.
- $\mathbf{K_D}$: is the derivative gain, which responds to the rate of change of the error, improving damping.

The PID controller balances the system's dynamic response by adjusting the error correction through three actions. The proportional term reacts immediately to the error, the integral term corrects for any accumulated bias over time, and the derivative term helps dampen the oscillations, making PID control ideal for systems requiring fast, accurate, and stable responses.

5.4.2 Application to a Double Joint System

In a double joint robotic system, PID control can be applied to regulate the motion of each joint independently, taking into account the position and velocity errors, as well as their accumulated discrepancies. The system uses two control inputs, τ_1 and τ_2, to generate the necessary torque to control each joint. The control equations for the joints are given by:

$$\tau_1 = K_{P1}(q_{1f} - u_3) + K_{I1} \int (q_{1f} - u_3)\, dt - K_{D1} u_5$$

$$\tau_2 = K_{P2}(q_{2f} - u_4) + K_{I2} \int (q_{2f} - u_4)\, dt - K_{D2} u_6$$

Where:

- $\mathbf{q_{1f}}$ and $\mathbf{q_{2f}}$: are the desired positions for joints 1 and 2, respectively.
- $\mathbf{u_3}$ and $\mathbf{u_4}$: represent the current positions of joints 1 and 2, respectively.
- $\mathbf{u_5}$ and $\mathbf{u_6}$: are the joint velocities of joints 1 and 2, respectively.
- $\mathbf{K_{P1}}, \mathbf{K_{I1}}, \mathbf{K_{D1}}$: are the proportional, integral, and derivative gains for joint 1, respectively;
- $\mathbf{K_{P2}}, \mathbf{K_{I2}}, \mathbf{K_{D2}}$: are the corresponding gains for joint 2.

These equations represent the control torque applied to each joint, with the proportional terms correcting immediate errors, the integral terms eliminating steady-state error, and the derivative terms damping the system's motion.

5.4.3 Dynamic Model of the System

The double joint system is subject to various dynamic effects, such as mass, inertia, Coriolis forces, and gravity, which must be accounted for in the control design. The general dynamic model of the system can be expressed as:

$$\ddot{q} = -M^{-1}(q)[C(q, \dot{q}) + G(q)] + \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

Where:

- $\mathbf{M}(\mathbf{q})$: is the mass/inertia matrix, dependent on the system's configuration.
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$: represents the Coriolis and centrifugal forces.
- $\mathbf{G}(\mathbf{q})$: captures the gravitational forces.

By substituting the PID control laws for τ_1 and τ_2, the resulting dynamic equations become:

$$\tau_1 = (M_1{+}M_2)L_1^2\,(K_{P1}(q_{1f} - u_3) + K_{I1}u_1 - K_{D1}u_5){+}M_2L_1L_2\cos(u_3{-}u_4)\,(K_{P2}(q_{2f} - u_4) + K_{I2}u_2 - K_I$$

$$\tau_2 = M_2L_1L_2\cos(u_3{-}u_4)\,(K_{P1}(q_{1f} - u_3) + K_{I1}u_1 - K_{D1}u_5){+}M_2L_2^2\,(K_{P2}(q_{2f} - u_4) + K_{I2}u_2 - K_{D2}u_6)$$

These equations describe how the applied torques τ_1 and τ_2 influence the system's joint positions and velocities, accounting for both the current state and accumulated error over time.

5.4.4 Discretization and Numerical Solution

To numerically solve the system, the differential equations are discretized by introducing the following state variables:

$$u_1 = q_1, \quad u_2 = q_2, \quad u_3 = \dot{q}_1, \quad u_4 = \dot{q}_2, \quad u_5 = \ddot{q}_1, \quad u_6 = \ddot{q}_2$$

The system is transformed into a set of first-order ordinary differential equations (ODEs):

$$\frac{dU}{dt} = H(t, U)$$

Where U is the state vector:

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix}$$

And $H(t, U)$ represents the system dynamics:

$$H(t, U) = \begin{bmatrix} u_3 \\ u_4 \\ u_5 \\ u_6 \\ \varphi(t, u_1, u_2, u_3, u_4, u_5, u_6) \\ \psi(t, u_1, u_2, u_3, u_4, u_5, u_6) \end{bmatrix}$$

This system can be solved using numerical methods such as ode45 in MATLAB, which computes the system's state trajectory over time, including the resulting torques τ_1 and τ_2.

5.4.5 Closed-Loop Transfer Function

The closed-loop transfer function of a PID-controlled system can be expressed as:

$$G(s) = \frac{(K_P + K_D s + \frac{K_I}{s})\omega_n^2}{s^3 + 2\zeta\omega_n s^2 + (\omega_n^2 + K_P)s + K_I}$$

Where:

- $\omega_\mathbf{n}$: is the natural frequency.
- ζ: is the damping ratio.
- $\mathbf{K_P}$, $\mathbf{K_D}$, and $\mathbf{K_I}$: are the proportional, derivative, and integral gains, respectively.

This transfer function describes the system's response to input commands. The proportional term influences the transient response, the derivative term controls damping and reduces oscillations, and the integral term ensures the system reaches the desired steady-state value.

5.4.6 Advantages of PID Control

PID controllers are highly versatile, allowing the tuning of proportional, integral, and derivative gains to achieve the desired dynamic response. The following are key advantages of PID control:

i) **Robustness**: By adjusting the three gains, various combinations of P, I, and D control can be achieved, making the controller adaptable to different system dynamics.

ii) **Error Elimination**: The integral term ensures that steady-state errors are eliminated, ensuring the system eventually reaches the target setpoint.

iii) **Oscillation Damping**: The derivative term helps in damping oscillations, providing smoother system behavior and improving transient response.

5.4.7 Disadvantages of PID Control

Despite its versatility, PID control has some drawbacks:

i) **Sensitivity to Noise**: The derivative term amplifies high-frequency noise, which can cause instability or oscillations if the input signal is noisy.

ii) **Difficulty with Non-linear Systems**: PID control may struggle with non-linear systems, where dynamic changes are significant, leading to degraded performance or instability.

5.4.8 Simulation

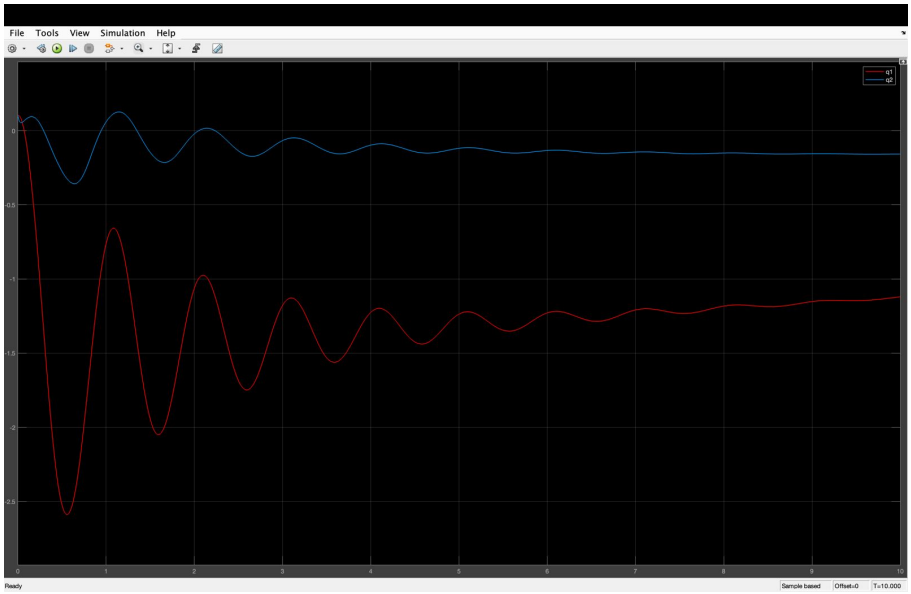


Figure 9: $K_P = 1$ and $K_I = 1$ and $K_D = 1$

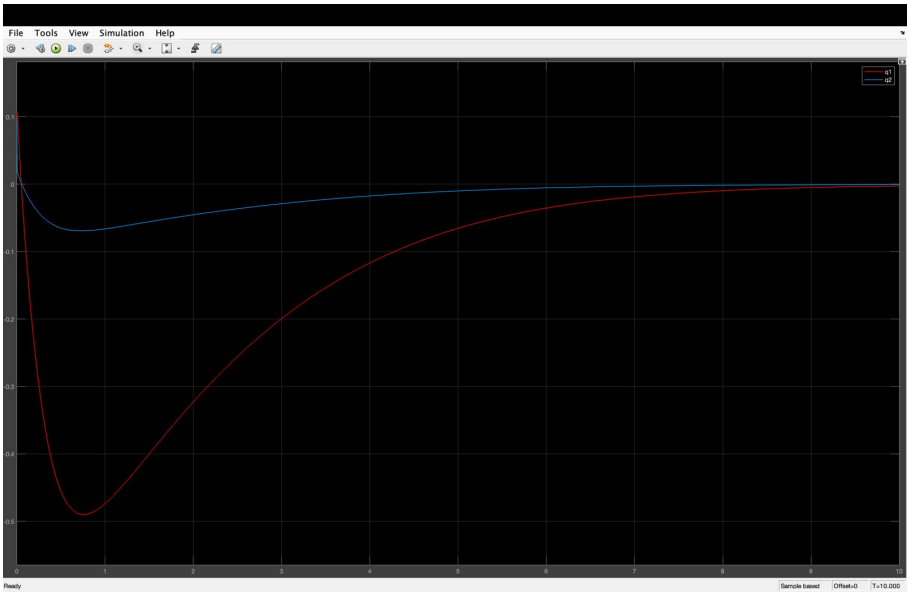


Figure 10: $K_P = 50$ and $K_I = 25$ and $K_D = 20$

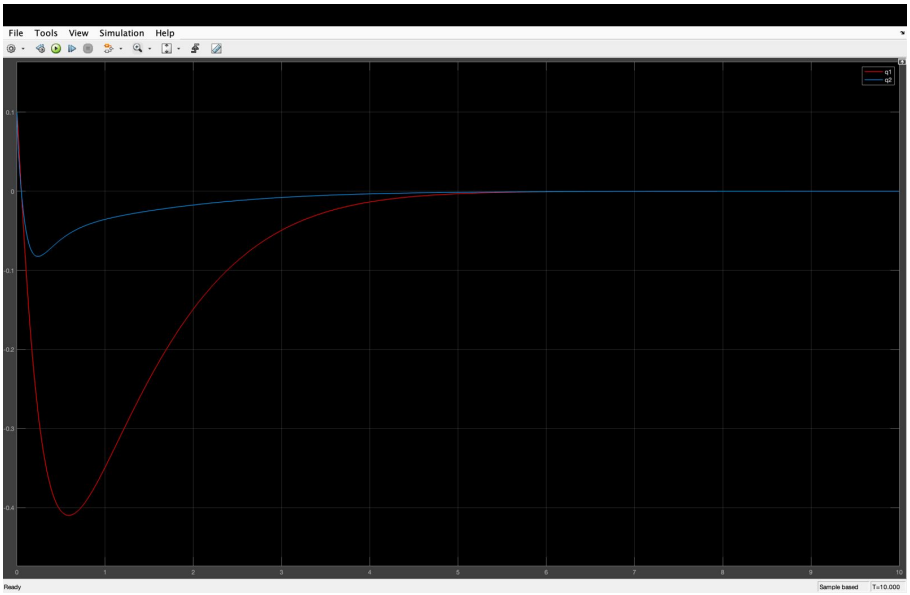


Figure 11: $K_P = 60$ and $K_I = 50$ and $K_D = 20$

6 Matlab Simulations

The system dynamics of a two-link manipulator can be solved using MATLAB's ode45 function, which is commonly used for solving ordinary differential equations (ODEs). It handles equations of the form:

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

where t is the independent variable, x is the vector of dependent variables, and f(t, x) is a function of t and x.

6.1 Code

Listing 1: Defining Variables

```
t_span = [0 20];
q0 = [0.1; 0.1];
qd0 = [0; 0];
x0 = [0; 0];
qd = [0; 0];
ode_func = @(t, s) dyn(t, s, qd);
[t, s] = ode45(ode_func, t_span, [q0; qd0; x0]);
q1 = s(:, 1);
q2 = s(:, 2);
q1d = s(:, 3);
q2d = s(:, 4);
x1 = s(:, 5);
x2 = s(:, 6);
```

The code begins by passing the initial conditions to ode45. The statespace function computes the derivatives of all states based on the current states, input, and desired angle, returning the derivative values in the matrix y. This function is repeatedly executed to obtain the state values over the specified time interval.

Listing 2: Matrices M, C, G and ode45 function

```
function func = dyn(t, s, qd)
    q1 = s(1);
    q2 = s(2);
    q1d = s(3);
    q2d = s(4);
    x1 = s(5);
    x2 = s(6);

    Kp1 = 50; Kd1 = 0; Ki1 = 5;
    Kp2 = 50; Kd2 = 0; Ki2 = 5;

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

    f1 = Kp1 * e1 + Ki1 * x1 - Kd1 * q1d;
    f2 = Kp2 * e2 + Ki2 * x2 - Kd2 * q2d;
    F = [f1; f2];

    m1 = 10; m2 = 5;
    l1 = 0.2; l2 = 0.1;

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

    tau = M * F;

    func = zeros(6, 1);
    func(1) = q1d;
    func(2) = q2d;
```

```
32      func(3) = - (5 * (tau(1) - (981 * cos(q1 + q2)) / 200 - (2943 * cos
            (q1)) / 100 + (q1d * q2d * sin(q2)) / 10 + (q2d * sin(q2) * (q1d
             + q2d)) / 10)) / (cos(q2)^2 - 3) ...
33                      - (5 * (2 * cos(q2) + 1) * ((sin(q2) * q2d^2) / 10 -
                            tau(2) + (981 * cos(q1 + q2)) / 200)) / (cos(q2)^2
                             - 3);
34      func(4) = (5 * (2 * cos(q2) + 1) * (tau(1) - (981 * cos(q1 + q2)) /
             200 - (2943 * cos(q1)) / 100 + (q1d * q2d * sin(q2)) / 10 + (
            q2d * sin(q2) * (q1d + q2d)) / 10)) / (cos(q2)^2 - 3) ...
35                    + (5 * (4 * cos(q2) + 13) * ((sin(q2) * q2d^2) / 10 -
                        tau(2) + (981 * cos(q1 + q2)) / 200)) / (cos(q2)^2 -
                        3);
36      func(5) = e1;
37      func(6) = e2;
38  end
```

6.2 P Control Simulation

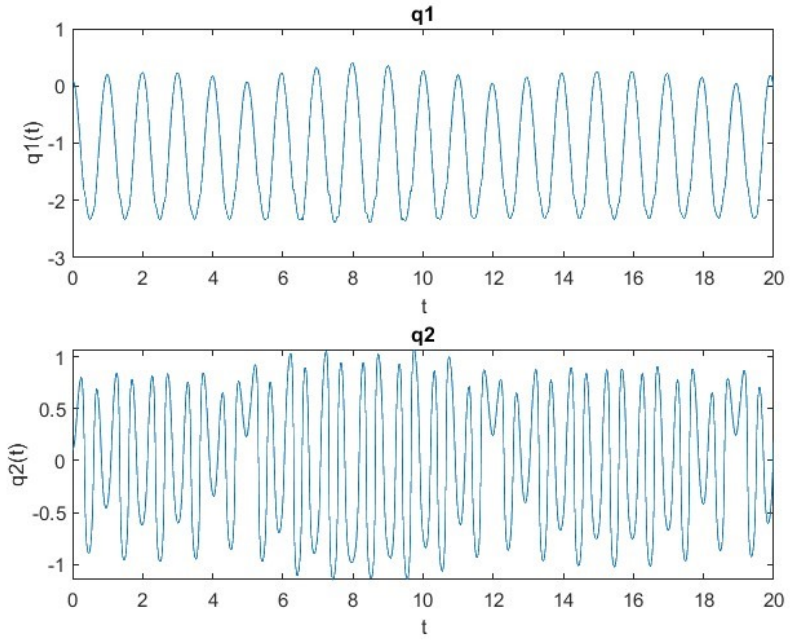


Figure 12: $K_P = 10$

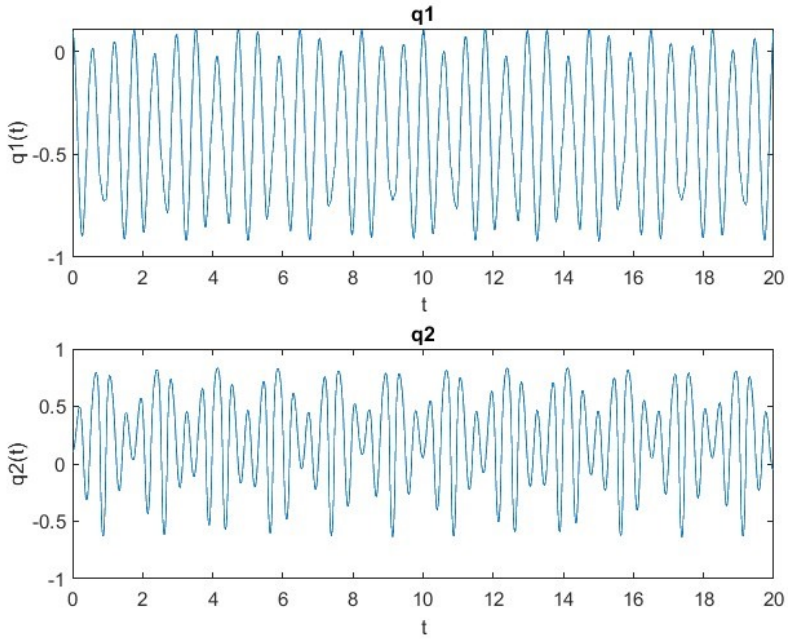


Figure 13: $K_P = 100$

6.3 PD Control Simulation

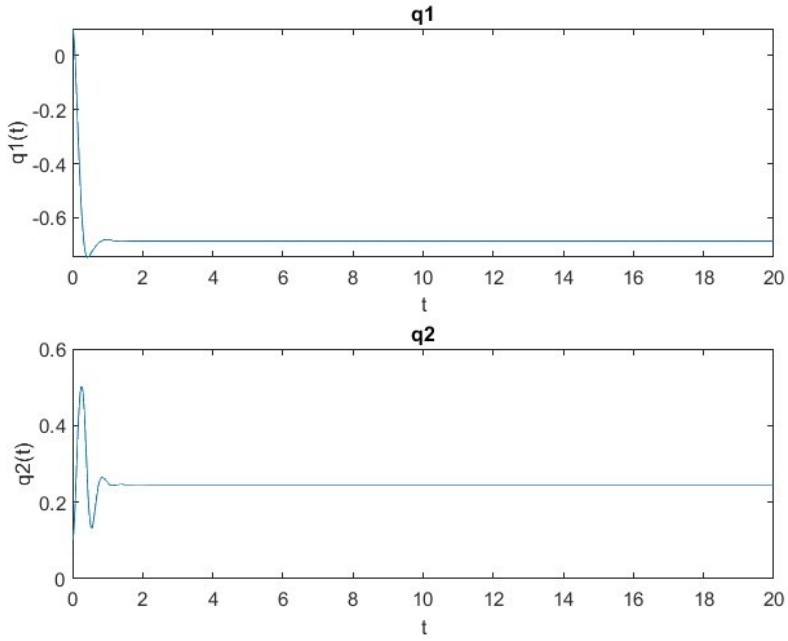


Figure 14: $K_P = 50$ and $K_D = 10$

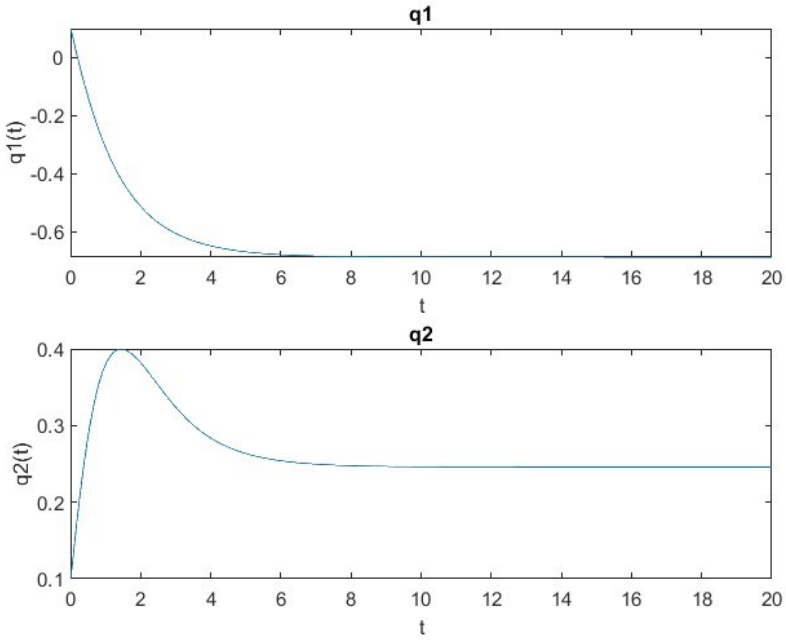


Figure 15: $K_P = 50$ and $K_D = 100$

6.4 PI Control Simulation

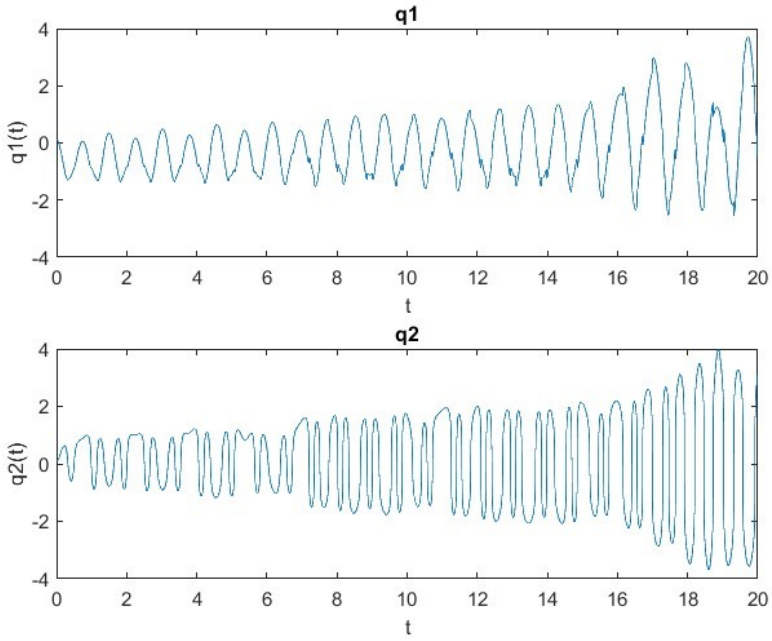


Figure 16: $K_P = 50$ and $K_I = 5$

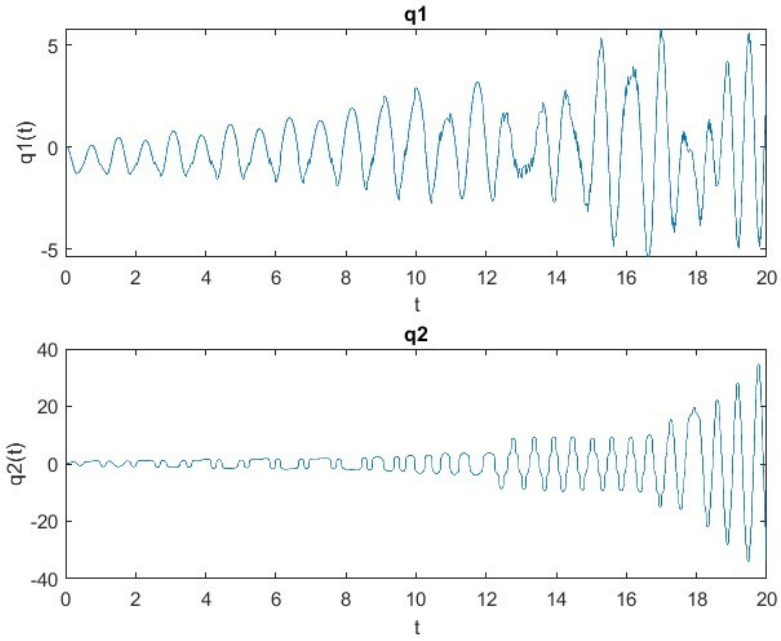


Figure 17: $K_P = 50$ and $K_I = 10$

6.5 PID Control Simulation

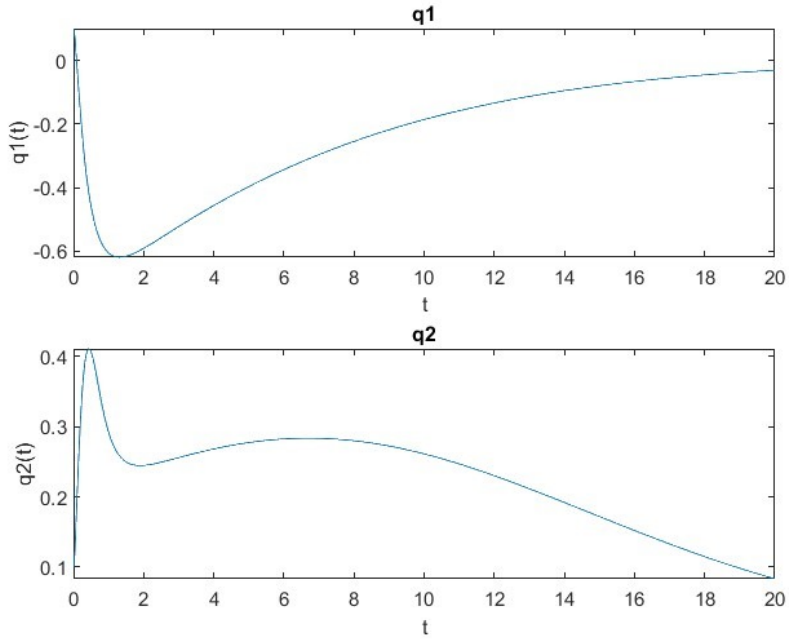


Figure 18: $K_P = 50$ and $K_D = 30$ and $K_I = 10$

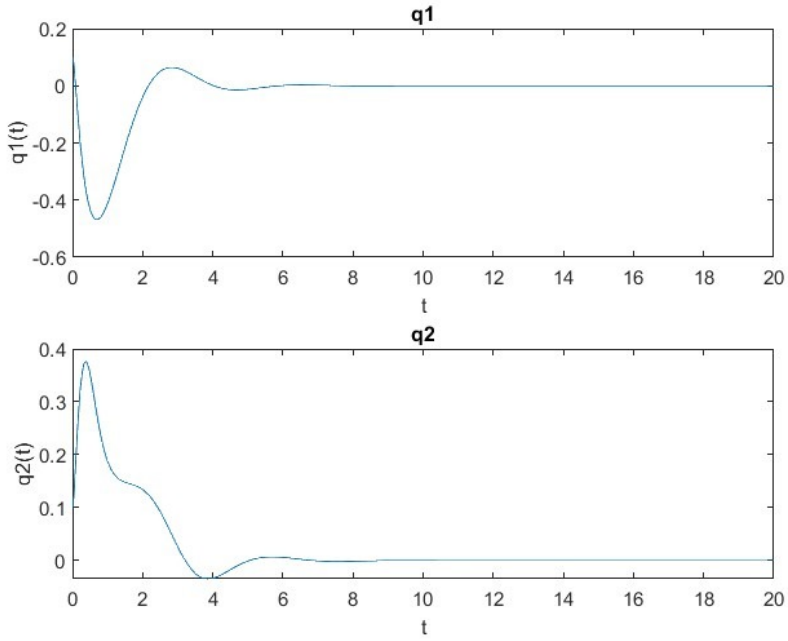


Figure 19: $K_P = 50$ and $K_D = 30$ and $K_I = 100$

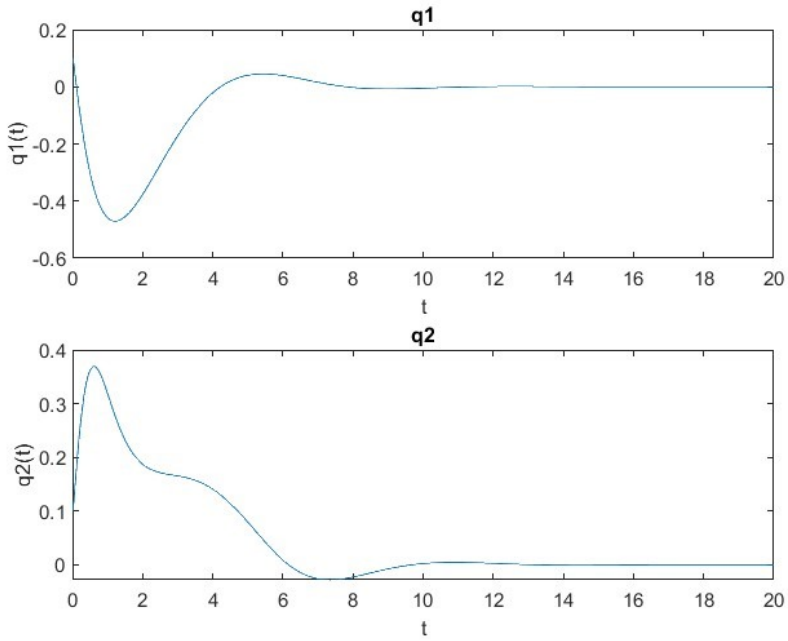


Figure 20: $K_P = 50$ and $K_D = 50$ and $K_I = 50$

7 Conclusion

The summary outlines the application of PI, PD, and PID controllers in controlling systems, such as a double-joint robotic mechanism:

- **PI Controller**: Focuses on immediate error correction and eliminating long-term error. While it may result in slower transient responses compared to PD, it excels at reducing steady-state error, making it suitable for precision-oriented systems.
- **PD Controller**: Optimizes transient response by accelerating the system towards the desired state and minimizing oscillations. It prioritizes speed and stability but may not fully eliminate steady-state error, making it ideal for fast-moving systems where perfect long-term accuracy isn't critical.
- **PID Controller**: Combines the benefits of both PI and PD control by allowing tuning of proportional, integral, and derivative gains. This provides a flexible solution balancing speed, accuracy, and stability. However, its sensitivity to noise and difficulty handling non-linear systems may require advanced control strategies for robust performance.

In all cases, discretizing the control system and solving it numerically ensures the necessary control signals are calculated effectively for driving the system with minimal error.