INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Systems Thinking Project

SYSTUMMM THINKERS

Two- Link Robot Manipulator

Madhur Kankane
Roll NO: 2024102061
madhur.kankane@students.iiit.ac.in

Het Selarka
Roll NO: 2024102031
het.selarka@students.iiit.ac.in

Shrenil Patel
Roll NO: 2024102066
patel.shaileshkumar@students.iiit.ac.in

Siddhant Gudwani
Roll NO: 2024102042
siddhant.gudwani@students.iiit.ac.in

Saumya Vira
Roll NO: 2024102044
saumya.vira@students.iiit.ac.in

Anish Toshniwal
Roll NO: 2024102009
anish.toshniwal@students.iiit.ac.in

1 Introduction

Robotics is an interdisciplinary field that combines mechanical engineering, electronics, computer science, and control theory to develop autonomous and semi-autonomous systems capable of interacting with and manipulating their environments. In industrial and research settings, robotic manipulators—particularly articulated arms—are widely used due to their precision, repeatability, and adaptability to a range of tasks, such as pick-and-place operations, welding, painting, and assembly.

Among these, the **two-link robotic manipulator** serves as a foundational model in robotics. It is mechanically simple but dynamically rich, making it ideal for studying control algorithms and dynamic modeling techniques. This manipulator consists of two rigid links connected via revolute joints, allowing motion within a plane and enabling two degrees of freedom. When modeled as a **double pendulum**, the system captures the complexity and nonlinearity inherent in multi-link robotic arms, making it a suitable platform for developing and validating advanced control strategies.

1.1 Conceptual Modeling

In this project, the two-link robotic arm is modeled as a planar double pendulum system. Each link is assumed to be a massless rod with a point mass at its end, though in practice, mass and inertia are distributed along the links. The first link is connected to a fixed base, while the second link is attached to the end of the first. Both joints are actuated, allowing controlled rotation in the vertical plane. Gravity influences the system, and the rotational motion of the links creates complex coupled dynamics.

This setup effectively captures the behavior of a simplified human arm or industrial robotic arm operating in a plane. Despite the low degrees of freedom, the model exhibits rich dynamic behavior—including oscillatory and even chaotic motion under certain conditions—which makes it valuable for exploring fundamental control problems in robotics.

1.2 Theoretical Framework

The dynamic equations governing the two-link manipulator are derived using the **Lagrangian formulation**, a method grounded in classical mechanics. This approach involves calculating the system's total kinetic energy (T) and potential energy (V), and forming the Lagrangian function $L = T - V$. The Euler-Lagrange equations are then applied to obtain the equations of motion for each joint.

This yields a system of coupled, second-order nonlinear differential equations that describe the evolution of the joint angles over time. These equations account for mass, link lengths, gravity, angular velocities, and torques applied at the joints. Due to their complexity, analytical solutions are typically not feasible, and numerical techniques are used to simulate the system.

1.3 Methodology

To control the manipulator's motion, we implement a **Proportional-Integral-Derivative (PID)** controller at each joint. The PID controller computes a torque based on the difference between the current joint angle and the desired angle. This error signal is processed using three terms:

- **Proportional (P):** Reacts to the magnitude of the error.
- **Integral (I):** Eliminates steady-state error over time.
- **Derivative (D):** Anticipates system behavior by considering the rate of change of the error.

Each joint is controlled independently using its own PID loop. Although the joints are dynamically coupled, independent control with careful tuning of PID gains is often sufficient for stabilization and trajectory tracking in planar systems.

The model and control strategy are implemented in **MATLAB**. Symbolic tools are used to derive the Lagrangian-based dynamic equations, which are then solved numerically using solvers such as `ode45`. The simulation environment allows for testing various initial conditions, disturbances, and PID parameter sets to analyze system performance.

1.4 Objectives

The primary objectives of this project are as follows:

1. **Modeling:** Develop a mathematical model of the two-link robotic manipulator using the Lagrangian formulation.
2. **Control Design:** Design and tune PID controllers to regulate joint motion and stabilize the system.
3. **Simulation:** Implement the dynamic model and control laws in MATLAB for simulation and performance analysis.
4. **Evaluation:** Analyze controller performance in terms of accuracy, stability, settling time, and robustness under varying conditions.

This study provides insight into how classical control techniques like PID can be effectively applied to nonlinear robotic systems. While the PID controller does not account for system coupling explicitly, with proper tuning it can achieve satisfactory results. The work also sets the stage for future exploration of more advanced control strategies, such as model-based control, adaptive control, or learning-based methods.

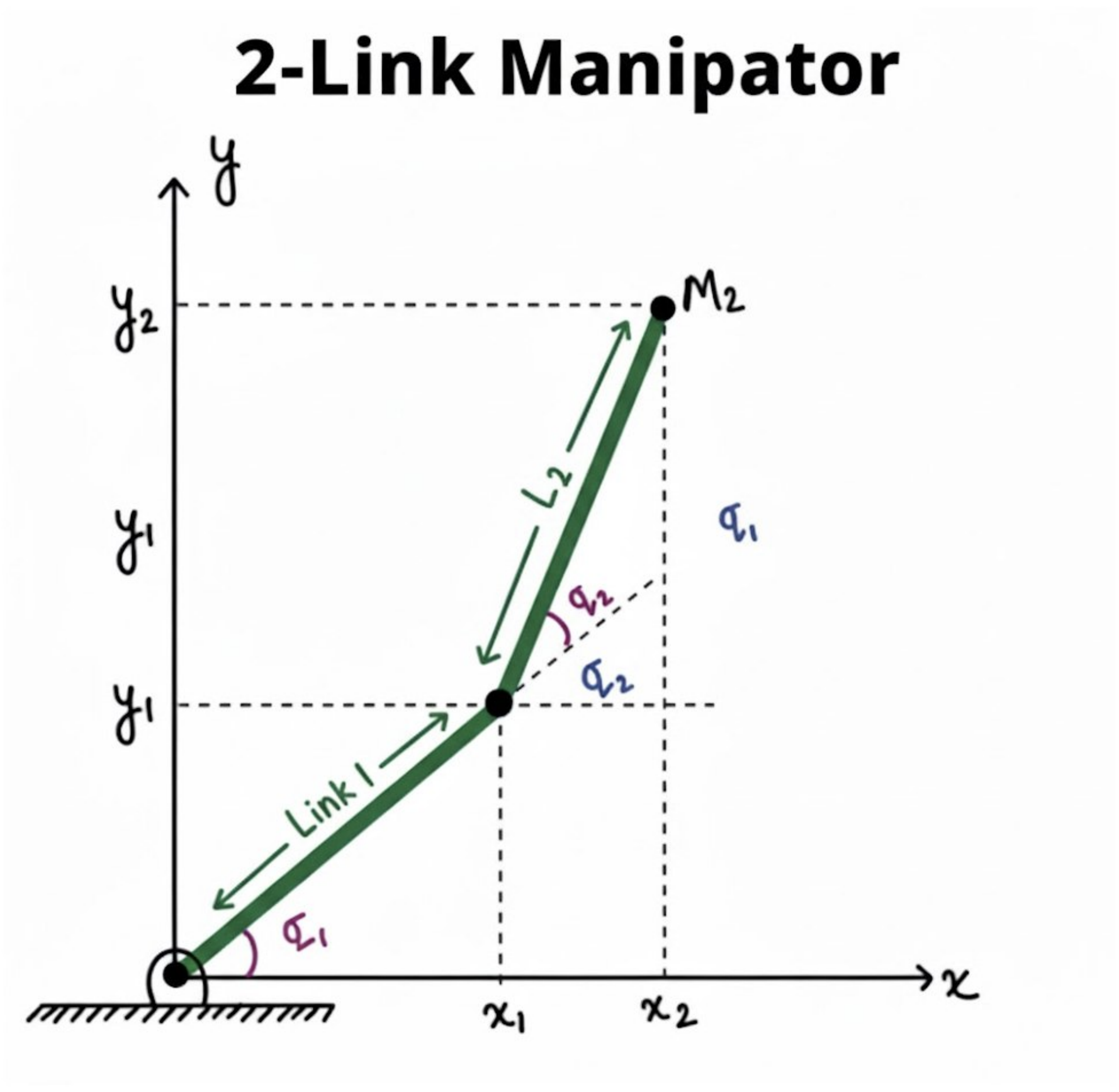


Figure 1: Diagram of the two-link manipulator.

2 Dynamics of Two Link Manipulators

The System consists of two-masses connected by rods. Considering the rods are ideal(i.e weightless). The bars have lengths L_1 and L_2. The Masses are denoted by M_1 and M_2 respectively. Let θ_1 and θ_2 denote the angles in which the first rod rotates about the Origin and second rod rotates about the endpoint of the first rod respectively. The System has two degrees of freedom θ_1 and θ_2.

The equations for the x-position and the y-position of M_1 are given by :

$$x_1 = L_1 \cos(\theta_1) \tag{2.1}$$
$$y_1 = L_1 \sin(\theta_1) \tag{2.2}$$

The equations for the x-position and the y-position of M_2 are given by :

$$x_2 = L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) \tag{2.3}$$
$$y_2 = L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) \tag{2.4}$$

We know that The velocity of Body is the rate of change of position vector of body with respective time. Considering position vector r:

$$r = x\hat{i} + y\hat{j}$$
$$\frac{dr}{dt} = \frac{dx}{dt}\hat{i} + \frac{dy}{dt}\hat{j}$$

The Magnitude of velocity of body is given by

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

So, we calculate the velocities of M_1 and M_2 using the following formulas:

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

The parameters represented in above two equations are :

$$(\dot{x}_1) = -L_1(\dot{\theta}_1)\sin(\theta_1) \tag{2.5}$$
$$(\dot{y}_1) = L_1(\dot{\theta}_1)\cos(\theta_1) \tag{2.6}$$
$$(\dot{x}_2) = -L_1(\dot{\theta}_1)\sin(\theta_1) - L_2(\dot{\theta}_2+\dot{\theta}_1)\sin(\theta_1+\theta_2) \tag{2.7}$$
$$(\dot{y}_2) = L_1(\dot{\theta}_1)\cos(\theta_1) + L_2(\dot{\theta}_2+\dot{\theta}_1)\cos(\theta_1+\theta_2) \tag{2.8}$$
$$\dot{\theta}_1 = \frac{d\theta_1}{dt} \tag{2.9}$$
$$\dot{\theta}_2 = \frac{d\theta_2}{dt} \tag{2.10}$$

2.1 Kinetic Energy (KE) Derivation

The total Kinetic Energy of the system is the sum of the kinetic energies of the two masses:

$$KE = KE_1 + KE_2 = \frac{1}{2}M_1v_1^2 + \frac{1}{2}M_2v_2^2$$

This can be expressed using the Cartesian velocity components:

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

First, let's find the squared speed v_1^2 for mass M_1:

$$\begin{aligned} v_1^2 &= \dot{x}_1^2 + \dot{y}_1^2 \\ &= (-L_1\dot{\theta}_1\sin\theta_1)^2 + (L_1\dot{\theta}_1\cos\theta_1)^2 \\ &= L_1^2\dot{\theta}_1^2\sin^2\theta_1 + L_1^2\dot{\theta}_1^2\cos^2\theta_1 \\ &= L_1^2\dot{\theta}_1^2(\sin^2\theta_1 + \cos^2\theta_1) \\ &= L_1^2\dot{\theta}_1^2 \end{aligned}$$

So, the kinetic energy of the first mass is $KE_1 = \frac{1}{2}M_1L_1^2\dot{\theta}_1^2$.

Next, we find the squared speed v_2^2 for mass M_2. We expand $\dot{x}_2^2$ and $\dot{y}_2^2$:

$$\begin{aligned}\dot{x}_2^2 &= \Big(-L_1\dot{\theta}_1\sin\theta_1 - L_2(\dot{\theta}_1+\dot{\theta}_2)\sin(\theta_1+\theta_2)\Big)^2 \\ &= L_1^2\dot{\theta}_1^2\sin^2\theta_1 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2\sin^2(\theta_1+\theta_2) \\ &\quad + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\sin\theta_1\sin(\theta_1+\theta_2)\end{aligned}$$

$$\begin{aligned}\dot{y}_2^2 &= \Big(L_1\dot{\theta}_1\cos\theta_1 + L_2(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_1+\theta_2)\Big)^2 \\ &= L_1^2\dot{\theta}_1^2\cos^2\theta_1 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2\cos^2(\theta_1+\theta_2) \\ &\quad + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos\theta_1\cos(\theta_1+\theta_2)\end{aligned}$$

Now, we sum them, grouping terms with common factors:

$$\begin{aligned}v_2^2 = \dot{x}_2^2 + \dot{y}_2^2 = {} & L_1^2\dot{\theta}_1^2(\sin^2\theta_1+\cos^2\theta_1) \\ & + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2(\sin^2(\theta_1+\theta_2)+\cos^2(\theta_1+\theta_2)) \\ & + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)[\cos\theta_1\cos(\theta_1+\theta_2)+\sin\theta_1\sin(\theta_1+\theta_2)]\end{aligned}$$

Using the identities $\sin^2\alpha + \cos^2\alpha = 1$ and $\cos(\alpha-\beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$, we simplify:

$$\begin{aligned}v_2^2 &= L_1^2\dot{\theta}_1^2(1) + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2(1) + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_1-(\theta_1+\theta_2)) \\ &= L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos(-\theta_2) \\ &= L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_2)\end{aligned}$$

So, the kinetic energy of the second mass is:

$$\begin{aligned}KE_2 &= \frac{1}{2}M_2\Big[L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos\theta_2\Big] \\ &= \tfrac{1}{2}M_2\Big(L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos\theta_2\Big)\end{aligned}$$

Finally, the total kinetic energy is $KE = KE_1 + KE_2$:

$$\boxed{KE = \tfrac{1}{2}M_1L_1^2\dot{\theta}_1^2 + \tfrac{1}{2}M_2\Big(L_1^2\dot{\theta}_1^2 + L_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2L_1L_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)\cos\theta_2\Big)} \tag{2.11}$$

2.2 Potential Energy (PE) Derivation

The general formula for Potential Energy is $PE = Mgh$, where h is the height above a reference point (e.g., the origin). The total PE is the sum of the potential energies of the two masses:

$$PE = PE_1 + PE_2$$

The height of mass M_1 is $h_1 = y_1 = L_1\sin(\theta_1)$.

$$PE_1 = M_1gh_1 = M_1gL_1\sin(\theta_1)$$

The height of mass M_2 is $h_2 = y_2 = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$.

$$PE_2 = M_2 g h_2 = M_2 g(L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2))$$

Summing them together gives the total potential energy:

$$\begin{aligned} PE &= PE_1 + PE_2 \\ &= M_1 g L_1 \sin(\theta_1) + M_2 g(L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)) \end{aligned}$$

This can also be written by grouping terms:

$$\boxed{PE = (M_1 + M_2) g L_1 \sin(\theta_1) + M_2 g L_2 \sin(\theta_1 + \theta_2)}$$

2.3 Lagrange Dynamics

Lagrangian mechanics uses energy-based principles, specifically the difference between kinetic and potential energy, to describe the dynamics of a system. This approach is particularly useful for complex systems with constraints, multi-body systems, and systems described in generalized coordinates.

- The Lagrangian is a function that summarizes the dynamics of the system. It is defined as: $\mathcal{L} = KE - PE$.
- Instead of using Cartesian coordinates, Lagrange dynamics employs generalized coordinates $q_1, q_2, q_3 \ldots$ that describe the configuration of the system. These can be angles, distances, or any other variables that uniquely define the system's configuration.
- The equations of motion are derived using the Lagrangian and take the following form for each generalized coordinate q_i:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \tag{2.12}$$

2.4 Computing Torque of both the bodies

This section details the derivation of the equations of motion by applying the Euler-Lagrange equation to the system's Lagrangian. The Lagrangian is given by:

$$\begin{aligned} \mathcal{L} = {} & \tfrac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \tfrac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 l_1 l_2 \cos(\theta_1 - \theta_2)\, \dot{\theta}_1 \dot{\theta}_2 \\ & - (m_1 + m_2) g l_1 \sin\theta_1 - m_2 g l_2 \sin\theta_2. \end{aligned} \tag{2.13}$$

(Note: This form of the Lagrangian and the subsequent equations are based on a convention where θ_2 is an absolute angle, not relative to the first link, which differs from the kinematics in Section 1.)

We compute the partial derivatives of $\mathcal{L}$ for each coordinate (θ_1, θ_2) and velocity $(\dot{\theta}_1, \dot{\theta}_2)$.

- **For coordinate θ_1:**

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (M_1 + M_2)L_1^2\dot{\theta}_1 + M_2L_1L_2\dot{\theta}_2\cos(\theta_1 - \theta_2)$$
$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - (M_1 + M_2)gL_1\cos(\theta_1)$$

- **For coordinate θ_2:**

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = M_2L_2^2\dot{\theta}_2 + M_2L_1L_2\dot{\theta}_1\cos(\theta_1 - \theta_2)$$
$$\frac{\partial \mathcal{L}}{\partial \theta_2} = M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - M_2gL_2\cos(\theta_2)$$

Now we substitute these into the Euler-Lagrange formula, $\frac{d}{dt}(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}) - \frac{\partial \mathcal{L}}{\partial \theta_i} = \tau_i$. This requires taking the total time derivative of the $\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}$ terms.

- **For τ_1 $(i = 1)$:**

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

Subtracting $\frac{\partial \mathcal{L}}{\partial \theta_1}$ gives the full expression for torque τ_1:

$$\boxed{\tau_1 = (M_1 + M_2)L_1^2\ddot{\theta}_1 + M_2L_1L_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) + M_2L_1L_2\dot{\theta}_2^2\sin(\theta_1 - \theta_2) + (M_1 + M_2)gL_1\cos(\theta_1)} \tag{2.14}$$

- **For τ_2 $(i = 2)$:**

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

Subtracting $\frac{\partial \mathcal{L}}{\partial \theta_2}$ gives the full expression for torque τ_2:

$$\boxed{\tau_2 = M_2L_2^2\ddot{\theta}_2 + M_2L_1L_2\ddot{\theta}_1\cos(\theta_1 - \theta_2) - M_2L_1L_2\dot{\theta}_1^2\sin(\theta_1 - \theta_2) + M_2gL_2\cos(\theta_2)}$$

The two torque equations form a system of two linear equations in terms of the accelerations $\ddot{\theta}_1$ and $\ddot{\theta}_2$. We can rewrite them by isolating the acceleration terms. Defining $\delta = \frac{M_2}{M_1+M_2}$, the equations become:

$$L_1\ddot{\theta}_1 + \delta L_1L_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) = \frac{\delta\tau_1}{M_2L_1} - \delta L_2\dot{\theta}_2^2\sin(\theta_1 - \theta_2) - g\cos(\theta_1) \tag{2.15}$$
$$L_2\ddot{\theta}_2 + L_1\ddot{\theta}_1\cos(\theta_1 - \theta_2) = \frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^2\sin(\theta_1 - \theta_2) - g\cos(\theta_2) \tag{2.16}$$

Solving this system of equations for $\ddot{\theta}_1$ and $\ddot{\theta}_2$ (e.g., using substitution or matrix inversion) gives the normal form of the dynamics equations:

$$\ddot{\theta}_1 = g_1(t, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2), \quad \ddot{\theta}_2 = g_2(t, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$$

where the explicit functions g_1 and g_2 are:

$$g_1 = \frac{\dfrac{\delta\tau_1}{M_2L_1} - \delta L_2\dot\theta_2^2\sin(\theta_1-\theta_2) - g\cos\theta_1 - \delta\cos(\theta_1-\theta_2)\left(\dfrac{\tau_2}{M_2L_2} + L_1\dot\theta_1^2\sin(\theta_1-\theta_2) - g\cos\theta_2\right)}{L_1\left(1-\delta\cos^2(\theta_1-\theta_2)\right)}, \tag{2.17}$$

$$g_2 = \frac{\dfrac{\tau_2}{M_2L_2} + L_1\dot\theta_1^2\sin(\theta_1-\theta_2) - g\cos\theta_2 - \cos(\theta_1-\theta_2)\left(\dfrac{\delta\tau_1}{M_2L_1} - \delta L_2\dot\theta_2^2\sin(\theta_1-\theta_2) - g\cos\theta_1\right)}{L_2\left(1-\delta\cos^2(\theta_1-\theta_2)\right)}. \tag{2.18}$$

2.5 First-Order State Form and Initial Conditions

To determine the angles θ_1 and θ_2, we must address the second-order system of ordinary differential equations specified above. The initial step involves transforming this system into a corresponding first-order ordinary differential equations system. We achieve this by defining four new variables:

$$\begin{aligned} u_1 &= \theta_1 \\ u_2 &= \theta_2 \\ u_3 &= \dot\theta_1 \\ u_4 &= \dot\theta_2 \end{aligned}$$

Upon differentiating, we obtain:

$$\begin{aligned} \dot u_1 &= u3 \\ \dot u_2 &= u4 \\ \dot u_3 &= g_1(t,u_1,u_2,u_3,u_4) \\ \dot u_4 &= g_2(t,u_1,u_2,u_3,u_4) \end{aligned}$$

Consequently, this yields a first-order nonlinear differential equation system as follows:

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

where the state vector U and the function vector S are:

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \quad S = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

with components:

$$\begin{aligned} s_1 &= u3 \\ s_2 &= u4 \\ s_3 &= g_1(t,u_1,u_2,u_3,u_4) \\ s_4 &= g_2(t,u_1,u_2,u_3,u_4) \end{aligned}$$

The initial conditions are given by:

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

where:

$$\begin{aligned} u_1(0) &= \theta_1(0) \\ u_2(0) &= \theta_2(0) \\ u_3(0) &= \dot{\theta}_1(0) \\ u_4(0) &= \dot{\theta}_2(0) \end{aligned}$$

We are taking the initial angles as:

$$\begin{aligned} \theta_1(0) &= 0.1 \quad [\text{rad}] \\ \theta_2(0) &= 0.1 \quad [\text{rad}] \\ \dot{\theta}_1(0) &= 0 \quad [\text{rad/s}] \\ \dot{\theta}_2(0) &= 0 \quad [\text{rad/s}] \end{aligned}$$

MATLAB's 'ode45' function can be employed to numerically determine the unknown vector U by solving the system of differential equations subject to the specified initial conditions.

3 State Model Equations

In control theory and system engineering, a state-space model is a mathematical representation of a dynamic system. It describes the system in terms of a set of first-order differential or difference equations, known as state equations. The state-space representation is a powerful and general method for modeling linear time-invariant systems. The Joint Torques are given by the Equation:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{3.1}$$

Since our system has 2 Degrees of Freedom then the expanded equation gives us:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} G_{11} \\ G_{12} \end{bmatrix}$$

For **Inertia Matrix** M(q)

- $M_{11} = m_2(l_1^2 + l_2^2) + m_1 l_1^2 + 2m_2 l_1 l_2 \cos(q_2)$
- $M_{12} = m_2 l_2 (l_1 + l_2 \cos(q_2))$
- $M_{22} = m_2 l_2^2$

For **Coriolis and centrifugal force matrix** $C(q, \dot{q})$

- $C_{11} = -m_2 l_1 l_2 \sin(q_2)\dot{q}_2$

- $C_{12} = -m_2 l_1 l_2 \sin(q_2)(\dot{q}_1 + \dot{q}_2)$
- $C_{21} = 0$
- $C_{22} = m_2 l_1 l_2 \sin(q_2)$

For **Gravity Vector** G(q)

- $G_{11} = m_1 g l_1 \cos(q_1) + m_2 g(l_2 \cos(q_1 + q_2) + l_1 \cos(q_1))$
- $G_{21} = m_2 g l_2 \cos(q_1 + q_2)$

The 2-link manipulator system has the following parameters for link 1 and link 2:

- Link 1: mass $m_1 = 10$ kg, length $l_1 = 0.2$ m
- Link 2: mass $m_2 = 5$ kg, length $l_2 = 0.1$ m

The gravitational acceleration is $g = 9.81$ m/s^2. The joint angles are initially at:

$$\begin{bmatrix} q_1(0) \\ q_2(0) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \text{rad}$$

The objective is to bring the joint angles to the target position:

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{rad}$$

This goal involves controlling the manipulator such that the angles converge from the initial state to the desired state over time. Rearranging the General Equations of Motion

$$\ddot{q} = -M(q)^{-1}C(q, \dot{q})\dot{q} - M(q)^{-1}G(q) + M(q)^{-1}\tau \tag{3.2}$$

$$\ddot{q} = -M^{-1}C\dot{q} - M^{-1}G + M^{-1}\tau \tag{3.3}$$

Calculating Inverse of Inertia Matrix

We know that for a 2×2 matrix A, given by:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The adjoint matrix and the determinant of matrix A are as shown below: The adjoint of A, denoted as Adj(A), is obtained by swapping the diagonal elements and changing the signs of the off-diagonal elements:

$$\text{Adj}(A) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

The determinant of matrix A, denoted as det(A), is given by:

$$\det(A) = ad - bc$$

The Inverse of matrix A, denoted as A^{-1}, is given by:

$$A^{-1} = \frac{1}{\det(A)}\mathrm{Adj}(A) = \frac{1}{\det(A)}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Therefore for Matrix M(q), determinant = det(M)

$$\det(M) = M_{11}M_{22} - M_{12}M_{12}$$

- $M_{11} = m_2(l_1^2 + l_2^2) + m_1 l_1^2 + 2m_2 l_1 l_2 \cos(q_2)$
- $M_{12} = m_2 l_2(l_1 + l_2\cos(q_2))$
- $M_{22} = m_2 l_2^2$

$$\det(M) = m_2 l_2^2(l_1^2 + l_2^2) + m_1 m_2 l_1^2 l_2^2 + 2m_2^2 l_1 l_2^3 \cos(q_2) - m_2^2 l_2^2 l_1^2 - m_2^2 l_2^4 \cos^2(q_2) - 2m_2^2 l_1 l_2^3 \cos(q_2) \tag{3.4}$$

$$\det(M) = m_2 l_2^2 l_1^2(m_1 + m_2\sin^2(q_2))$$

The Inverse of Inertia Matrix :

$$M^{-1} = \frac{1}{\det(M)}\begin{bmatrix} M_{22} & -M_{12} \\ -M_{12} & M_{11} \end{bmatrix}$$

We can solve for some theoretical values of forces given certain initial inputs. Solving for $\ddot{\theta}$, we get:

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

where

$$\tau_b = -M^{-1}(q)\tau.$$

Thus, we decoupled the system to have the new input:

$$\tau_b = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}.$$

However, the physical torque inputs to the system are given by

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

Let us denote the error signals by

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

where the target position of M_1 and M_2 are given by the angles θ_{1f} and θ_{2f}, respectively.

4 Mathematical Model For Robotic Manipulator

Controlling robotic manipulators poses significant challenges, particularly when it comes to stabilizing them at a precise, stationary position. This section primarily addresses using the computed torque control method to achieve the desired position of a robotic manipulator. After deriving the equation of motion, MATLAB is employed for control simulation. In this discussion, we will develop algorithms for PD, PI, and PID Controller Designs. "P" represents Proportional control. "I" denotes Integral control, and "D" stands for Derivative control. The algorithm operates by defining an error variable $V_{error} = V_{set} - V_{sensor}$, which represents the difference between the desired position (V_{set}) and the current position (V_{sensor}). The proportional term of the PID control is obtained by multiplying a defined constant K_P by the error. The integral term is derived by multiplying a constant K_I by the integral of the error over time. The derivative term is defined as a constant K_D multiplied by the derivative of the error with respect to time. Below is a table illustrating PID control:

Table 1: PID Control Terms and Their Effects on Control Systems

Term	**Math Function**	**Effect on Control System**
P	$K_p V_{error}$	Typically the main drive in a control loop, K_p reduces a large part of the overall error.
I	$K_i \int V_{error} dt$	Reduces the final error in a system. Summing even a small error over time produces a drive signal large enough to move the system toward a smaller error.
D	$K_D \frac{dV_{error}}{dt}$	Counteracts the K_p and K_i terms when the output changes quickly. This helps reduce overshoot and ringing. It has no effect on the final error.

From now on, θ is represented by 'q'.

$$\tau = M(q)\dot{q} + C(q, \dot{q})\dot{q} + G(q) \tag{4.1}$$

where,

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

$M_{11} = (m_1 + m_2)l_1^2 + m_2 l_2(l_2 + 2l_1 \cos(q_2))$, $M_{12} = m_2 l_2(l_2 + l_1 \cos(q_2))$. $M_{22} = m_2 l_2^2$

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

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

We can solve for some theoretical values of forces given certain initial inputs. Solving for $\dot{q}$ we get:

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

Thus, we decoupled the system to have the new input:

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

However, the physical torque inputs to the system are:

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

Let us denote the error signals by:

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

where the target position of M_1 and M_2 are given by the angles θ_{1f} and θ_{2f}, respectively. We assume that the system has initial positions:

$$q_0 = \begin{bmatrix} q_1(0) \\ q_2(0) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

A common technique for controlling a system with input is to use the following general structure of PID controller:

$$f = K_p e + K_d \dot{e} + K_i \int e dt$$

In our situation, the technique for controlling the double pendulum system with inputs f_1 and f_2 is to employ two independent controllers, one for each link, as follows:

$$f_1 = K_{p1}e_1(q_1) + K_{D1}\dot{e}_1(q_1) + K_{I1}\int e_1(q_1)dt = K_{p1}(q_{1f} - q_1) - K_{D1}\dot{q}_1 + K_{I1}\int (q_{1f} - q_1)dt \tag{4.2}$$

$$f_2 = K_{p2}e_2(q_2) + K_{D2}\dot{e}_2(q_2) + K_{I2}\int e_2(q_2)dt = K_{p2}(q_{2f} - q_2) - K_{D2}\dot{q}_2 + K_{I2}\int (q_{2f} - q_2)dt \tag{4.3}$$

where q_{1f} and q_{2f} are given constants. The complete system of equations with control is then:

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

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

We would like to emphasize that the actual physical torques are:

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

To implement the PID controller, we introduce the following new states:

$$x_1 = \int e(q_1)dt, \quad x_2 = \int e(q_2)dt$$

Differentiating with respect to t gives:

$$\dot{x}_1 = e(q_1) = q_{1f} - q_1 \quad \dot{x}_2 = e(q_2) = q_{2f} - q_2$$

The complete equations are:

$$\dot{x}_1 = q_{1f} - q_1 \quad \dot{x}_2 = q_{2f} - q_2$$

$$\begin{bmatrix} \tilde{q}_1 \\ \tilde{q}_2 \end{bmatrix} = -M^{-1}(q)[C(q,\dot{q})\dot{q} + G(q)] + \begin{bmatrix} K_{p1}(q_{1f} - q_1) - K_{d1}\dot{q}_1 + K_{I1}x_1 \\ K_{p2}(q_{2f} - q_2) - K_{d2}\dot{q}_2 + K_{I2}x_2 \end{bmatrix} \tag{4.4}$$

$$\tau = M(q) \begin{bmatrix} K_{p1}(q_{1f} - q_1) - K_{d1}\dot{q}_1 + K_{I1}x_1 \\ K_{p2}(q_{2f} - q_2) - K_{d2}\dot{q}_2 + K_{I2}x_2 \end{bmatrix} \tag{4.5}$$

In order to discretize the aforementioned system of differential equations over time, we convert them into a set of first-order ordinary differential equations by introducing six new variables defined as follows:

$$u_1 = \dot{q}_1 \qquad u_2 = \dot{q}_2, \qquad u_3 = \dot{q}_1 \qquad u_4 = \dot{q}_2$$
$$u_5 = x_1, \qquad u_6 = x_2$$

$$\begin{aligned}
\dot{u}_1 &= \dot{q}_1 = u_3, \\
\dot{u}_2 &= \dot{q}_2 = u_4, \\
\dot{u}_3 &= \dot{q}_1 + \phi(t, u_1, u_2, u_3, u_4, u_5, u_6), \\
\dot{u}_4 &= \dot{q}_2 + \psi(t, u_1, u_2, u_3, u_4, u_5, u_6), \\
\dot{u}_5 &= \dot{x}_1 - \dot{q}_1 = -u_1, \\
\dot{u}_6 &= \dot{x}_2 - \dot{q}_2 = -u_2
\end{aligned}$$

where ϕ and ψ are expressed in terms of u_k, $k = 1 - 6$, as

$$\begin{bmatrix} \phi \\ \psi \end{bmatrix} = -M^{-1}(q)[C(q,\dot{q})\dot{q} + G(q)] + \begin{bmatrix} K_{P1}(q_1 - u_1) - K_{D1}\dot{u}_3 + K_{I1}u_5 \\ K_{P2}(q_2 - u_2) - K_{D2}\dot{u}_4 + K_{I2}u_6 \end{bmatrix}$$

and $q = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

5 Controllers and their Impact on System Performance

It can be noted that within sections 3.1.2 and 3.2.1, the transfer functions take on a second-order structure. As a result, we carry out the evaluation for these two setups in a broad manner by designating the factor associated with the s^2 component as α and the factor linked to the s component as β.

In other words,

$$G(s) = \frac{1}{\alpha s^2 + \beta s}$$

Consequently, the transfer function for the closed-loop system, denoted as $T(s)$, may be written as:

$$T(s) = \frac{1}{\alpha s^2 + \beta s + 1} \tag{5.0.1}$$

Additionally, we recognize the standard structure for a system of second order:

$$T_{\text{general}}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.0.2}$$

In this way, we examine the $T(s)$ presented in equation 5.0.1 alongside the closed-loop transfer functions derived from implementing various types of controllers. We pay particular attention to the denominators because, in the context of a second-order setup, the numerator simply represents a scaled version of the natural frequency term ω_n.

5.1 Proportional Control

The standard schematic representation featuring the proportional controller, characterized by the gain K_p, is illustrated as follows:

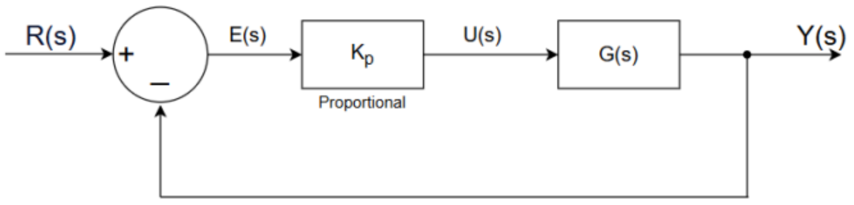


Figure 2: Block Diagram with a Proportional Controller

Thus, the transfer function for the closed-loop system, $T_P(s)$, when incorporating a proportional controller becomes:

$$\frac{Y(s)}{R(s)} = \frac{K_pG(s)}{1 + K_pG(s)}$$

$$\therefore T_P(s) = \frac{K_p}{\alpha s^2 + \beta s + K_p} \tag{5.1.1}$$

By contrasting the denominators from equations 5.0.1 and 5.1.1,

In the case of the proportional controller, the value of the natural frequency ω_{nP} is expressed as:

$$\omega_{nP} = \sqrt{\frac{K_p}{\alpha}} > 1$$

Furthermore, the formula for the damping ratio ζ_P turns out to be:

$$\zeta_P = \frac{\beta}{2\omega_{nP}}$$

$$\zeta_P = \frac{\beta}{2\sqrt{\frac{K_p}{\alpha}}} < \frac{\beta}{2\omega_n} \qquad \because \omega_n = 1 < \sqrt{\frac{K_p}{\alpha}}$$

It is evident that there has been a shift in the natural frequency, and should K_p exceed 1, then ω_n would rise accordingly. To keep the coefficient for the s term unchanged under these circumstances, the damping ratio ζ must reduce. Such a change leads to amplified and quicker vibrations, resulting in increased overshoot levels as an outcome.

5.2 Proportional Derivative Control

Now, let's suppose we incorporate a derivative component as well, featuring a gain K_d for the derivative action. The schematic for the PD controller is depicted below:

This leads us to,

$$\frac{Y(s)}{R(s)} = \frac{(K_p + K_d s)}{\alpha s^2 + \beta s + (K_p + K_d s)}$$

Hence, the closed-loop transfer function for this configuration, $T_{PD}(s)$, is provided by:

$$T_{PD}(s) = \frac{(K_p + K_d s)}{\alpha s^2 + (\beta + K_d)s + K_p} \tag{5.2.1}$$

In order to evaluate the influence that adding a derivative action has on how the system behaves, we refer to equations 5.1.1 and 5.2.1. It becomes apparent that the damping ratio ζ_{PD} experiences an enhancement, whereas the natural frequency stays consistent. Therefore, introducing the derivative action serves solely to boost the damping ratio. This occurs because the derivative mechanism acts in a predictive manner, helping to mitigate any potential rise in error during the transient phase of the response.

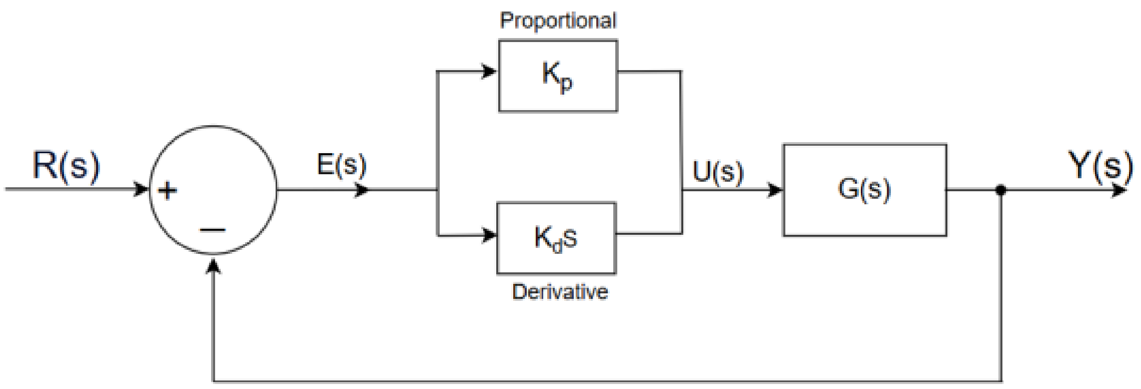


Figure 3: Block Diagram with a PD Controller

5.2.1 Practical Usage of Derivative Control

Within a PID controller framework (which stands for Proportional-Integral-Derivative), the filtering parameter is generally tied to the derivative action component. The standard equation governing PID control is expressed as:

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

In this context:

- K_p represents the gain for the proportional part.
- K_i denotes the gain for the integral part.
- K_d indicates the gain for the derivative part.
- $e(t)$ is the error value, calculated as the disparity between the target setpoint and the measured actual value.

The derivative component tends to be vulnerable to noise at high frequencies since the process of differentiation can intensify such disturbances. To counteract this issue, filtering is commonly applied to the derivative. The transfer function for a typical low-pass filter is given by:

$$H(s) = \frac{1}{1 + \frac{s}{\omega_0}}$$

Here, ω_0 signifies the filter's cutoff frequency.

The parameter known as the filter coefficient (often labeled as N) helps in refining the derivative component through the integration of a low-pass filter. The adjusted form of the derivative term becomes:

$$D(s) = \frac{K_d s}{1 + s\frac{1}{N}}$$

In this expression:

- s serves as the operator in the Laplace domain.
- N acts as the filter coefficient, dictating the extent of filtering employed. A higher N value leads to less filtering, thereby enhancing the system's sensitivity to variations, whereas a smaller N provides greater smoothing but may slow down the overall response.

To encapsulate, the role of the filter coefficient N is to diminish the derivative term's vulnerability to noise through the incorporation of a low-pass filtering mechanism.

5.3 Proportional Integral Control

At this point, the controller module includes an integral controller characterized by the gain K_i, alongside the proportional controller maintaining the identical gain K_p as previously established.

The schematic representation for the PI controller is shown below:

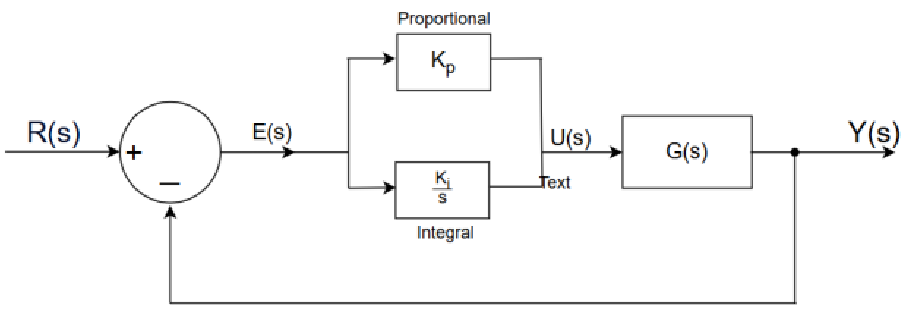


Figure 4: Block Diagram with a PI Controller

$$\frac{Y(s)}{R(s)} = \frac{G'(s)}{1 + G'(s)}$$

In this case, $G'(s)$ represents the transfer function of the open-loop system.

$$\because G'(s) = (K_i + K_p s)\frac{1}{s(\alpha s^2 + \beta s)}$$

$$\therefore \frac{Y(s)}{R(s)} = \frac{(K_i + K_p s)}{s(\alpha s^2 + \beta s) + (K_i + K_p s)}$$

Thus, the closed-loop transfer function for this setup, $T_{PI}(s)$, is stated as:

$$T_{PI}(s) = \frac{(K_i + K_p s)}{\alpha s^3 + \beta s^2 + K_p s + K_i}$$

It is obvious that this expression reaches a third-order level. Consequently, direct comparison to the second-order model defined by parameters ω_n and ζ is not feasible. Nevertheless, it is reasonable to conclude that the influences from each distinct element will remain noticeable and contribute significantly to how the system responds overall.

5.4 Proportional Integral Derivative Control

Currently, we are equipped with a PID controller, where the respective gains for proportional, derivative, and integral actions are denoted as K_p, K_d, and K_i.

The schematic for the PID controller is presented below:

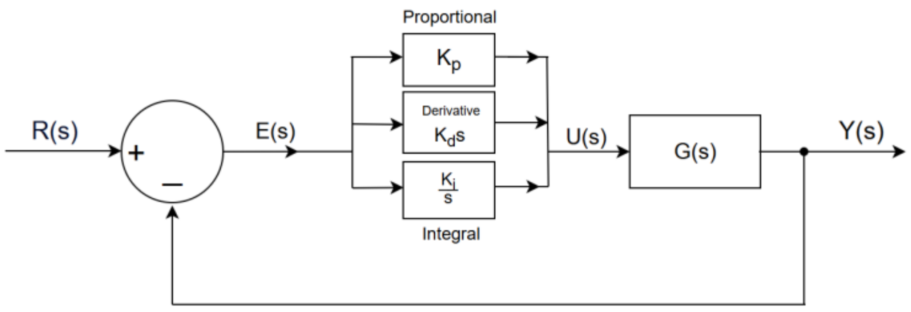


Figure 5: Block Diagram with PID

In this scenario,

$$\because G'(s) = (K_i + K_p s + K_d s^2)\frac{1}{s(\alpha s^2 + \beta s)}$$

$$\therefore \frac{Y(s)}{R(s)} = \frac{(K_i + K_p s + K_d s^2)}{s(\alpha s^2 + \beta s) + (K_i + K_p s + K_d s^2)}$$

The closed-loop transfer function for this arrangement, $T_{PID}(s)$, is therefore:

$$T_{PID}(s) = \frac{(K_d s^2 + K_p s + K_i)}{\alpha s^3 + (\beta + K_d)s^2 + K_p s + K_i}$$

Much like the case with Proportional-Integral control, this function also attains a third-order status. Hence, conducting an analysis here proves challenging. However, based on the graphical representations derived from simulations, it can be observed that the level of damping achieved with PID is superior when compared to that of PI control.

5.5 Steady State analysis

Next, we evaluate the performance of the system equipped with the aforementioned controllers concerning their errors in the steady state. The application of the Final Value Theorem is valid solely when the entity under consideration (in this instance, $sE(s)$) possesses poles that all have negative real components.

We presume that the polynomial in the denominator, which could be of degree 1, 2, or 3 (given that the most complex scenario among the controllers involves third-order systems), features poles exclusively with negative real parts to guarantee system stability. For deriving the steady-state error specifically in this part, there will be a minor adjustment in the symbols used, as outlined below.

- $G(s)$ denotes the plant's transfer function.
- $C(s)$ signifies the controller's transfer function.

$$G(s) = \frac{1}{\alpha s^2 + \beta s}$$

This serves as a universal transfer function applicable to systems dealing with both θ and z.

- $T(s)$ represents the forward-path transfer function, formed by multiplying the plant's and controller's transfer functions. ($T(s) = G(s)C(s)$)

Moreover, we employ unity feedback, so $H(s) = 1$. The corresponding block diagram using this symbolism is displayed below:

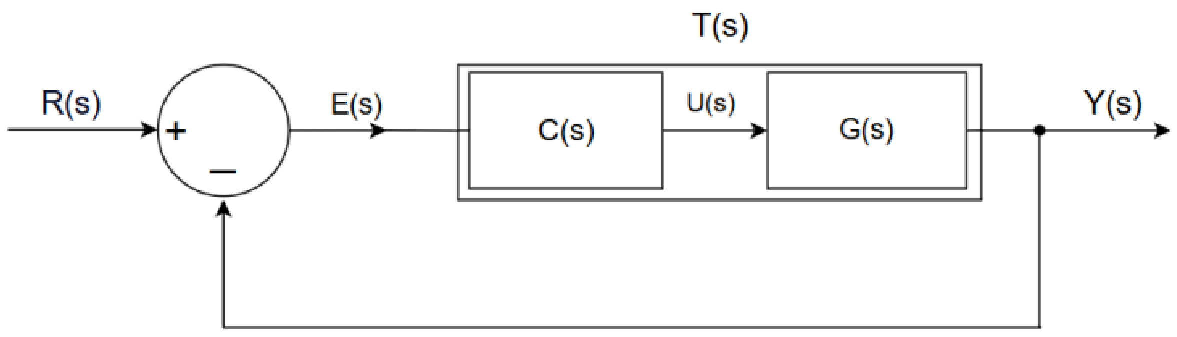


Figure 6: Block Diagram

Consequently, the closed-loop transfer function is,

$$\frac{Y(s)}{R(s)} = \frac{T(s)}{1 + T(s)H(s)} = \frac{T(s)}{1 + T(s)}$$

However, $Y(s) = E(s)T(s)$

$$E(s) = \frac{R(s)}{1 + T(s)}$$

Utilizing the final value theorem, we arrive at:

$$\lim_{t \to \infty} e(t) = \lim_{s \to 0} sE(s) = \lim_{s \to 0} \frac{sR(s)}{1 + T(s)}$$

Given that the input reference is merely a scaled step function (with scaling constant k), having a Laplace transform of $\frac{1}{s}$, it follows that $sR(s) = k$.

This yields the limiting expression as:

$$\lim_{s \to 0} \frac{k}{1 + T(s)} \tag{5.5.1}$$

We proceed to employ equation 5.5.1 for determining the steady-state values across various controllers.

5.5.1 Proportional (K_p) Controller

Accordingly, we find:

$$T(s) = \frac{K_p}{\alpha s^2 + \beta s}$$

The limiting value of $sE(s)$ (derived from 5.5.1) computes to:

$$\lim_{s \to 0} \frac{k}{1 + \frac{K_p}{\alpha s^2 + \beta s}} = \lim_{s \to 0} k \frac{\alpha s^2 + \beta s}{\alpha s^2 + \beta s + K_p} = 0$$

This limit resolves to 0 since the expression reaches a definite value as s approaches 0. Therefore, $e_{SS} = 0$ in the scenario involving a proportional controller.

This outcome may appear at odds with the common notion that proportional controllers exhibit steady-state errors. Yet, in our particular system, the error truly diminishes to zero because there exists no force proportional to the linear or angular displacement. Essentially, the absence of a constant in the denominator eliminates any contribution to persistent error within the system.

5.5.2 Proportional-Derivative (PD) (K_p, K_d) Controller

The expression for $T(s)$ becomes:

$$T(s) = \frac{K_p + K_d s}{\alpha s^2 + \beta s}$$

From equation 5.5.1, the limit for $sE(s)$ results in:

$$\lim_{s\to 0} \frac{k}{1+\frac{K_p+K_d s}{\alpha s^2+\beta s}} = \lim_{s\to 0} k \frac{\alpha s^2+\beta s}{\alpha s^2+(\beta+K_d)s+K_p} = 0$$

Hence, this limit also amounts to 0, given that the expression attains a definite form as s nears 0. As such, $e_{SS} = 0$ for the PD controller scenario too.

5.5.3 Proportional-Integral (PI) (K_p, K_i) Controller

$T(s)$ computes to:

$$T(s) = \left(K_p + \frac{K_i}{s}\right) \frac{1}{\alpha s^2+\beta s} = \frac{sK_p+K_i}{\alpha s^3+\beta s^2}$$

Utilizing 5.5.1, the limit of $sE(s)$ yields:

$$\lim_{s\to 0} \frac{k}{1+\frac{sK_p+K_i}{\alpha s^3+\beta s^2}} = \lim_{s\to 0} k \frac{\alpha s^3+\beta s^2}{\alpha s^3+\beta s^2+K_p s+K_i} = 0$$

This limit similarly resolves to 0, as the expression takes on a definite shape when s approaches 0. Consequently, $e_{SS} = 0$ for the PI controller as well.

5.5.4 PID (K_p, K_i, K_d) controller

$T(s)$ is determined as:

$$T(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) \frac{1}{\alpha s^2+\beta s} = \frac{K_d s^2+sK_p+K_i}{\alpha s^3+\beta s^2}$$

From 5.5.1, the limit for $sE(s)$ computes to:

$$\lim_{s\to 0} \frac{k}{1+\frac{K_d s^2+sK_p+K_i}{\alpha s^3+\beta s^2}} = \lim_{s\to 0} k \frac{\alpha s^3+\beta s^2}{\alpha s^3+(\beta+K_d)s^2+K_p s+K_i} = 0$$

Thus, the steady-state error associated with the PID controller reduces to zero, meaning $e_{SS} = 0$ for the PID Controller.

6 Analysis of Controller Performance Under Different Tuning Philosophies

This report presents a comprehensive analysis of Proportional-Integral (PI), Proportional-Derivative (PD), and Proportional-Integral-Derivative (PID) controllers applied to a two-link robotic manipulator. The primary objective is to evaluate the performance of each controller under three distinct tuning philosophies—Balanced, Aggressive, and Conservative—by examining key performance metrics such as settling time, percentage overshoot, and steady-state error.

6.1 Balanced (Normal) Controller Tuning

This strategy aims for an optimal compromise between speed, stability, and accuracy. The tuning provides a clear middle ground. The PID controller, in particular, achieves a fast settling time (65.234 ms) while completely eliminating overshoot in one joint, showcasing its versatility.

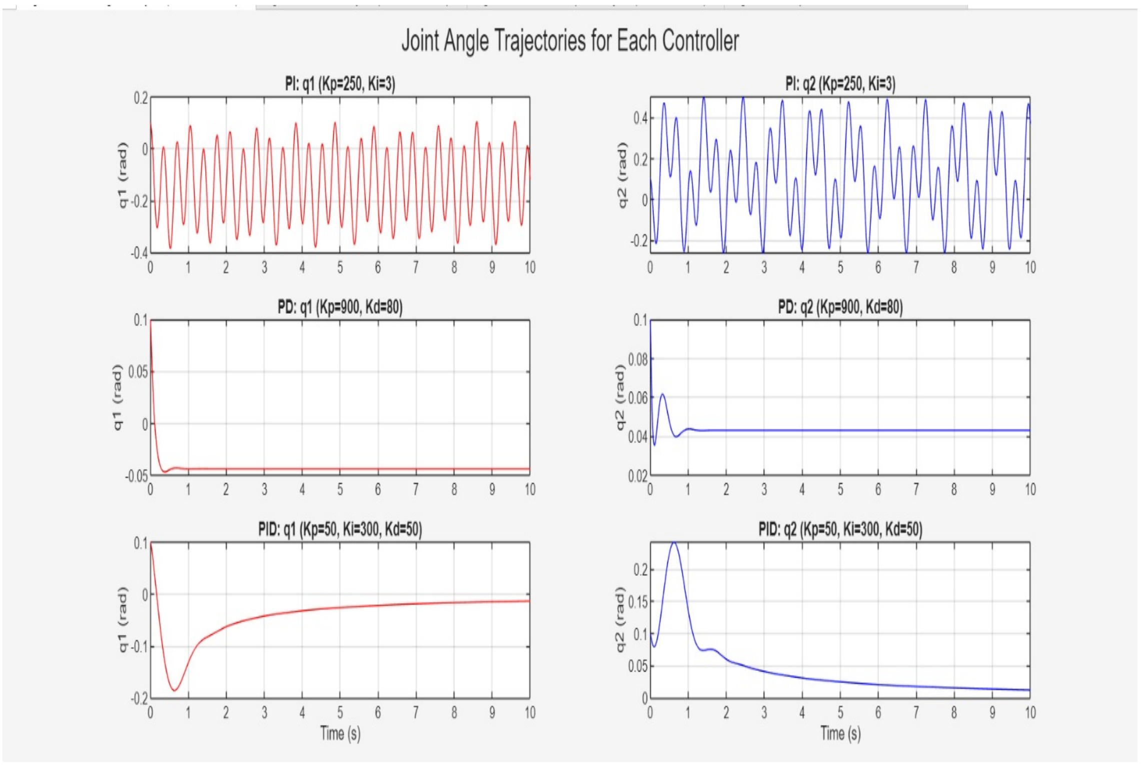


Figure 7: Balanced Tuning - Plot 1.

- **PI ($K_p = 250, K_i = 3$):** Unstable. Massive overshoot (382.94%) and fails to settle, proving its unsuitability.
- **PD ($K_p = 900, K_d = 80$):** Good stability. Eliminates overshoot for q2 (0.00%) with moderate overshoot for q1 (46.37%). However, it leaves a noticeable steady-state error (0.0435), limiting its precision.

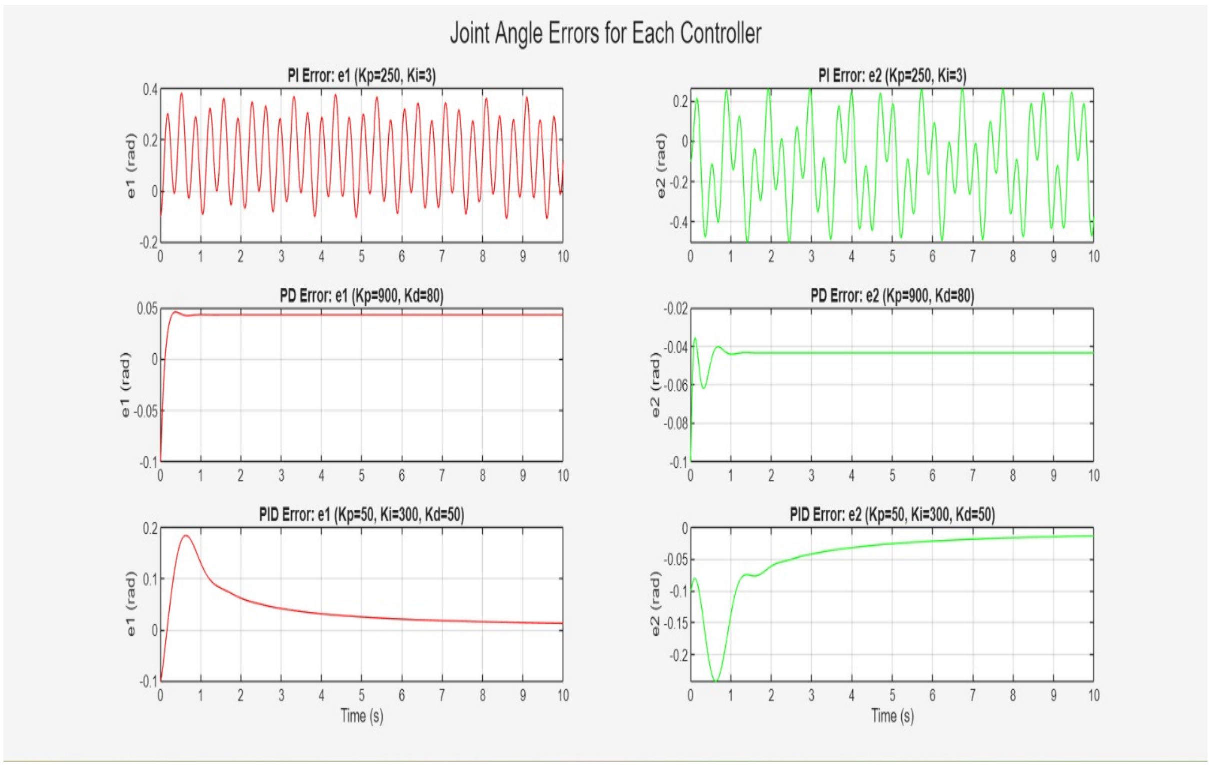


Figure 8: Balanced Tuning - Plot 2.

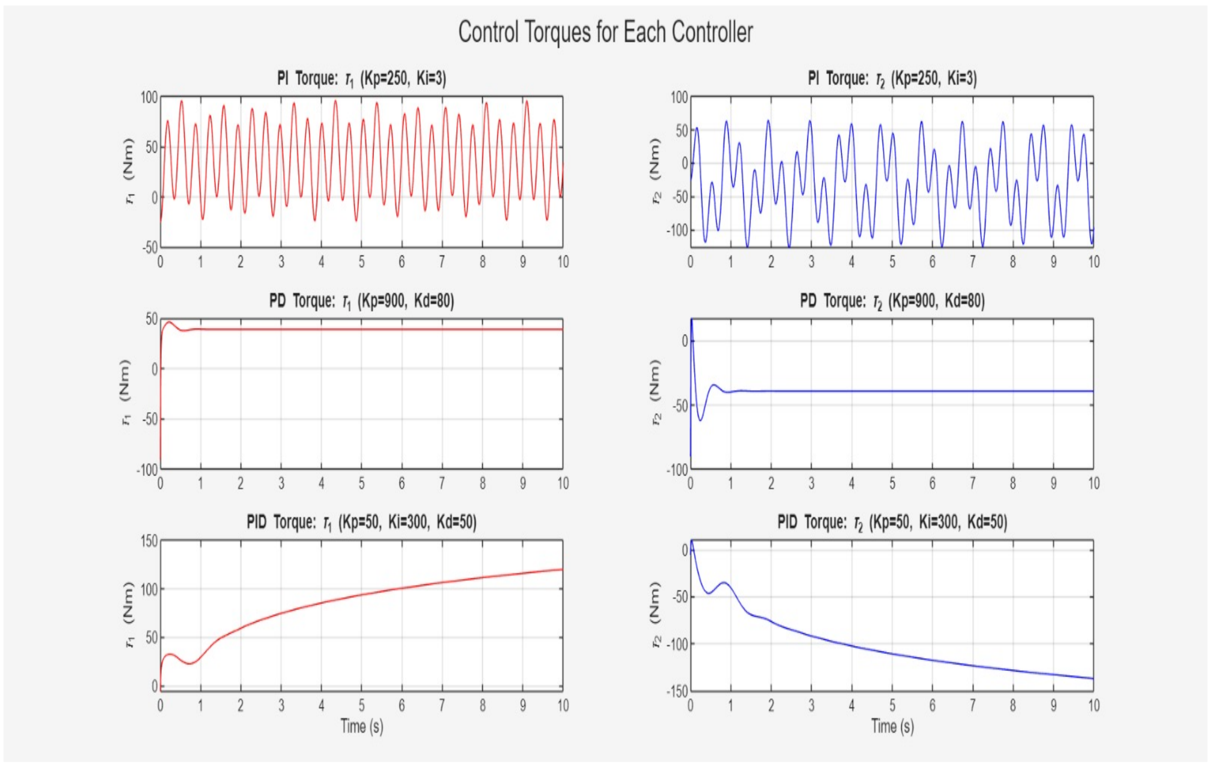


Figure 9: Balanced Tuning - Plot 3.

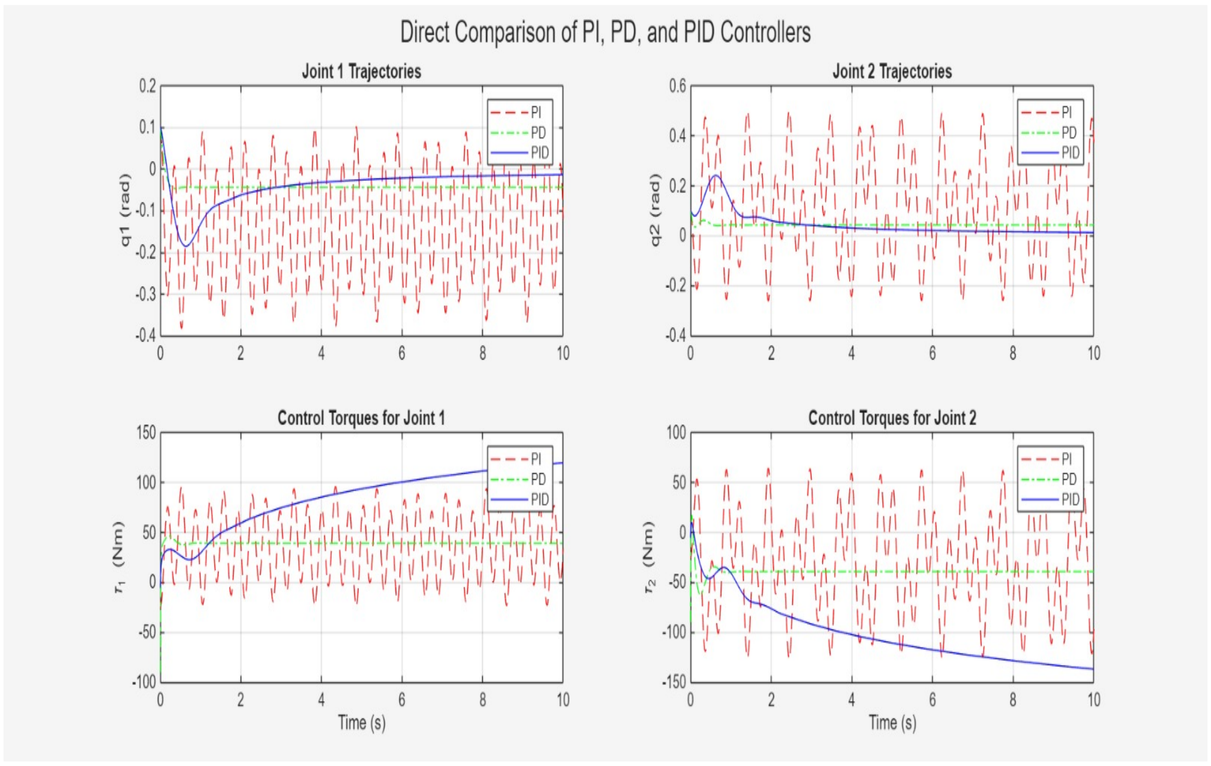
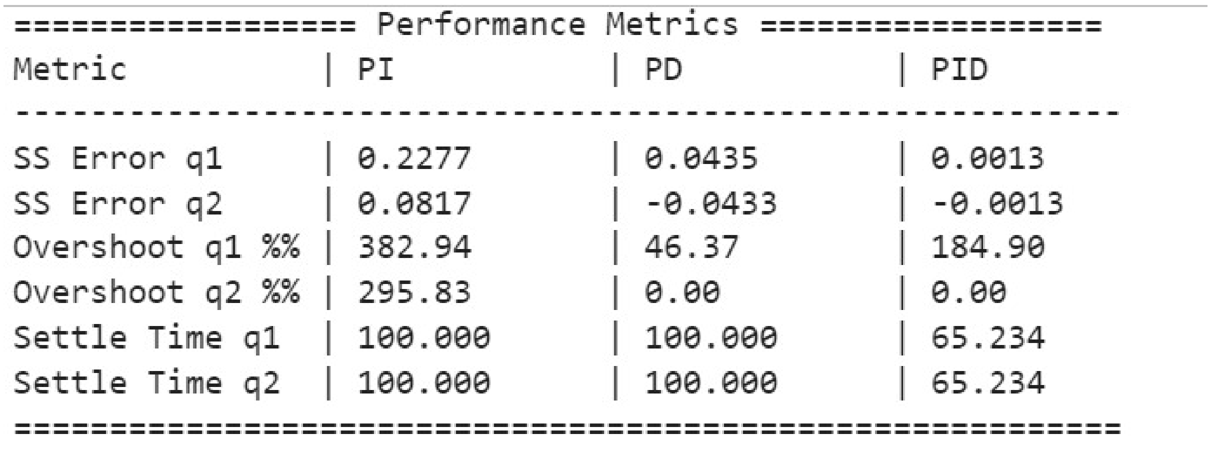


Figure 10: Balanced Tuning - Plot 4.



```
================== Performance Metrics ==================
Metric             | PI              | PD              | PID
-----------------------------------------------------------

SS Error q1        | 0.2277          | 0.0435          | 0.0013
SS Error q2        | 0.0817          | -0.0433         | -0.0013
Overshoot q1 %%    | 382.94          | 46.37           | 184.90
Overshoot q2 %%    | 295.83          | 0.00            | 0.00
Settle Time q1     | 100.000         | 100.000         | 65.234
Settle Time q2     | 100.000         | 100.000         | 65.234
===========================================================
```

Figure 11: Balanced Tuning - Plot 5.

- **PID ($K_p = 50, K_d = 50, K_i = 300$):** Optimal Performance. It is fast (65.234 ms), stable (zero overshoot for q2), and perfectly accurate (zero steady-state error). This represents the best overall choice for general-purpose tasks.

6.2 Aggressive Controller Tuning

This strategy prioritizes the fastest possible response time, accepting overshoot and instability as trade-offs. This tuning philosophy pushes the system to its limits. The PID controller achieves the fastest settling time of any scenario (48.549 ms), but this speed comes at the cost of significant overshoot.

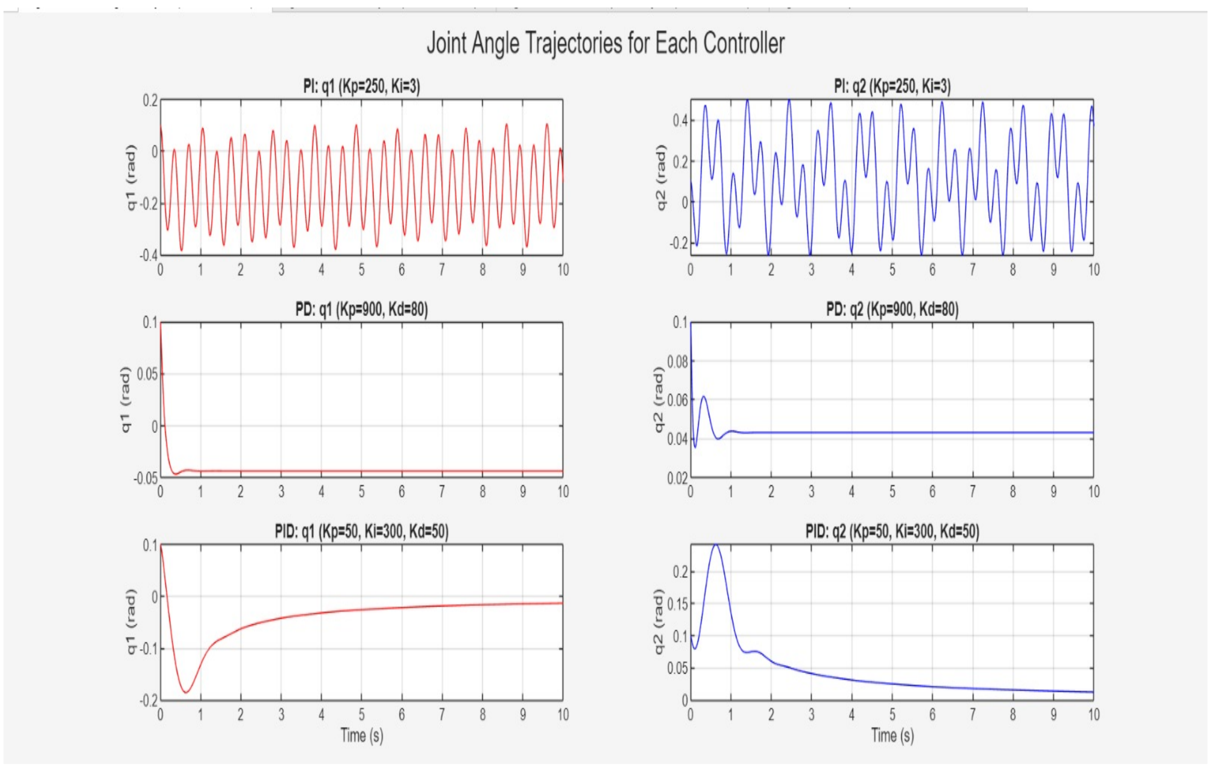


Figure 12: Aggressive Tuning - Plot 1.

- **PI ($K_p = 800, K_i = 50$):** Highly unstable. Extreme overshoot (193.43% and 217.88%) and fails to settle. This confirms PI is a poor choice for dynamic systems.
- **PD ($K_p = 1200, K_d = 40$):** Fast but unstable. The insufficient damping leads to significant overshoot (41.30%) and a persistent steady-state error (0.0327).
- **PID ($K_p = 200, K_d = 20, K_i = 400$):** Fastest Performance. It achieves the quickest settling time but suffers from very high overshoot (140.40%). It remains perfectly accurate. This configuration is only suitable for applications where speed is the absolute priority and overshoot is acceptable.

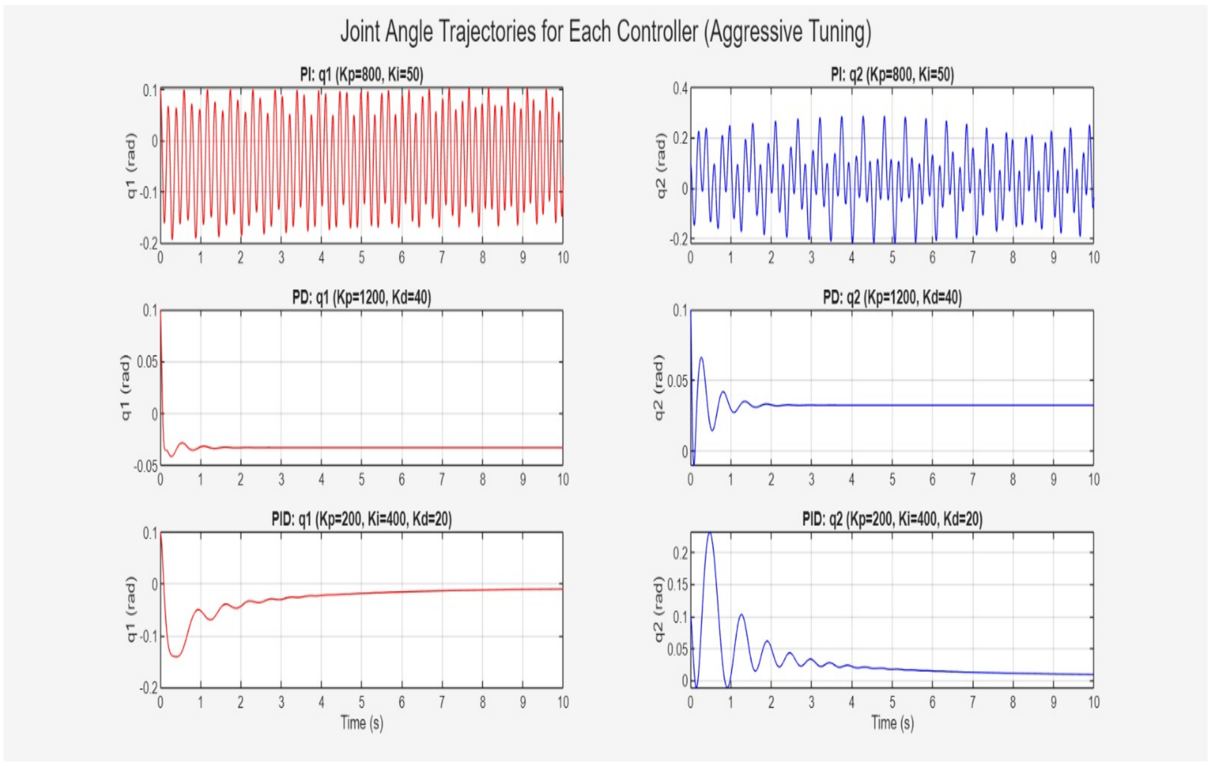


Figure 13: Aggressive Tuning - Plot 2.

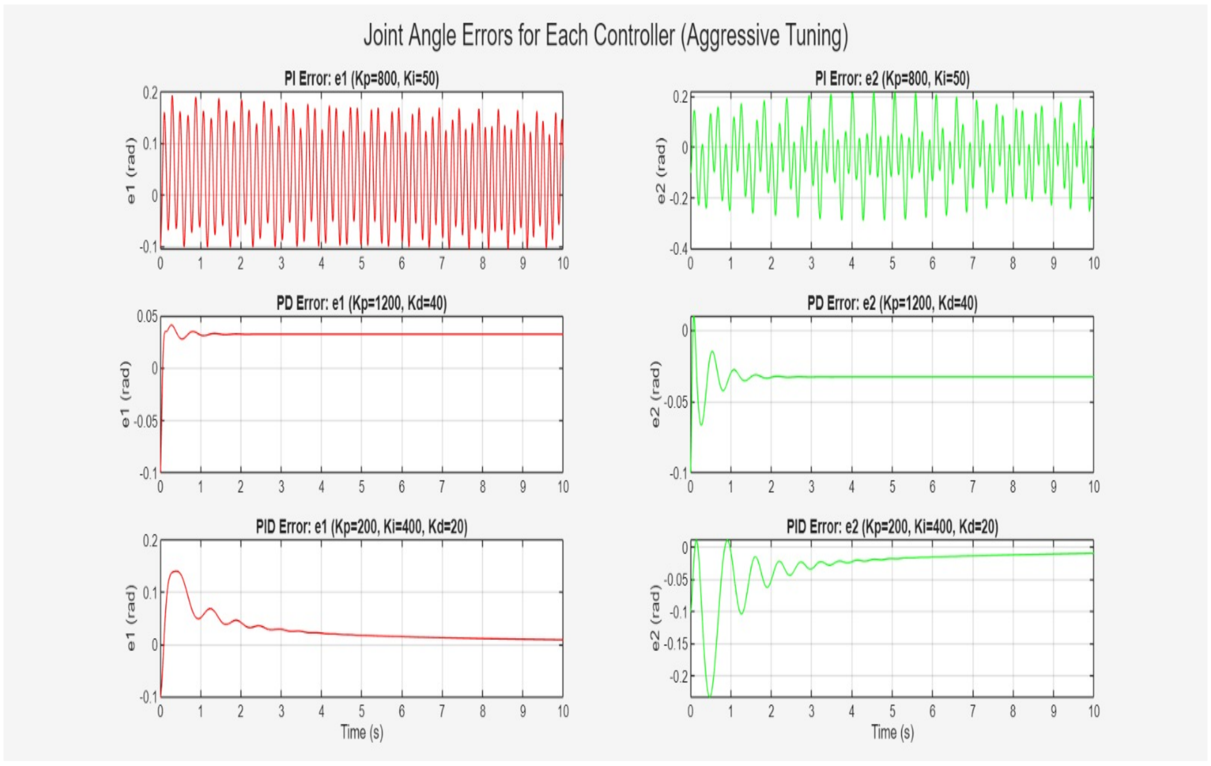


Figure 14: Aggressive Tuning - Plot 3.

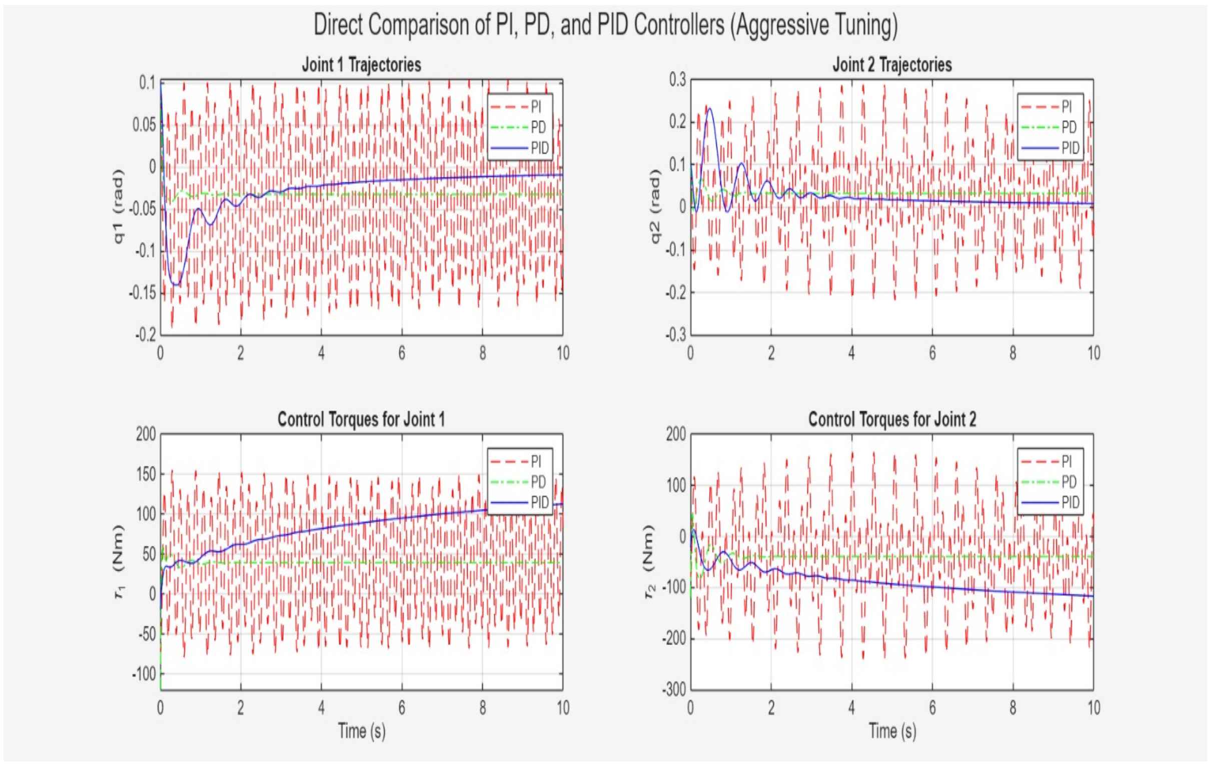
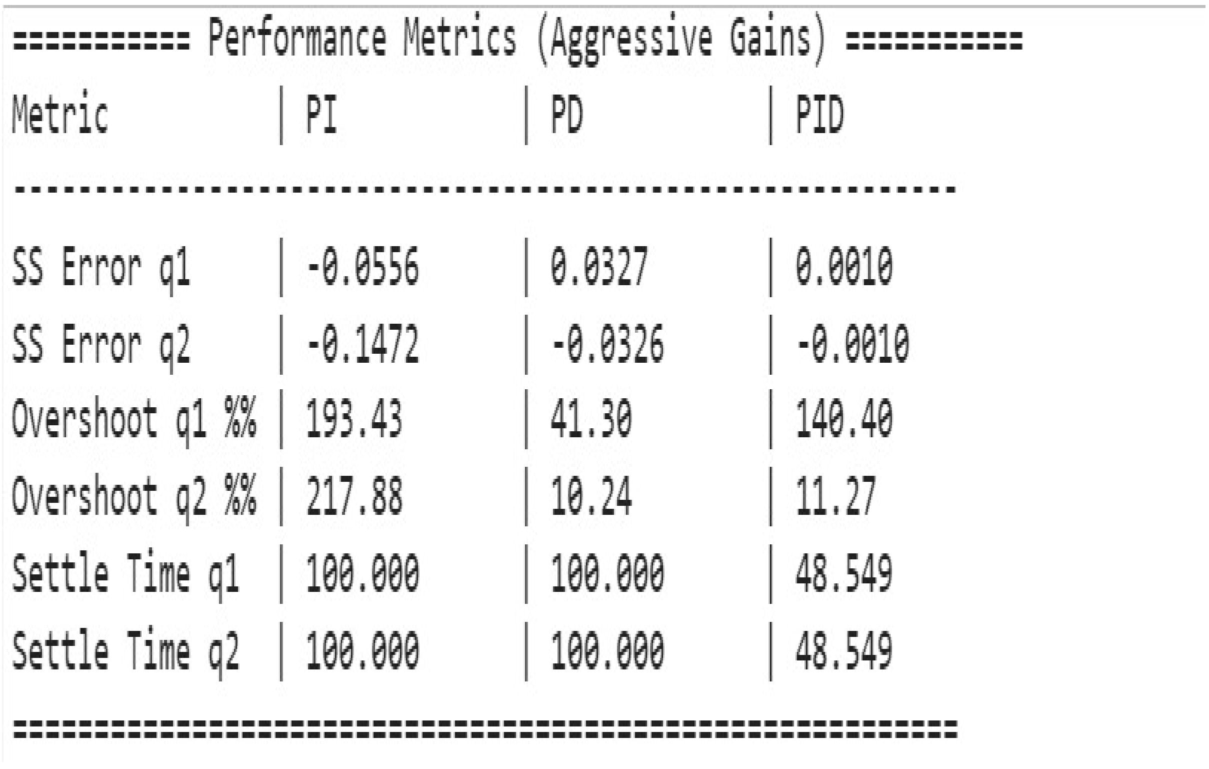


Figure 15: Aggressive Tuning - Plot 4.



```
=========== Performance Metrics (Aggressive Gains) ===========
Metric          | PI          | PD          | PID
----------------------------------------------------------
SS Error q1     | -0.0556     | 0.0327      | 0.0010
SS Error q2     | -0.1472     | -0.0326     | -0.0010
Overshoot q1 %% | 193.43      | 41.30       | 140.40
Overshoot q2 %% | 217.88      | 10.24       | 11.27
Settle Time q1  | 100.000     | 100.000     | 48.549
Settle Time q2  | 100.000     | 100.000     | 48.549
==========================================================
```

Figure 16: Aggressive Tuning - Plot 5.

6.3 Conservative Controller Tuning

This strategy prioritizes maximum stability and smoothness, sacrificing speed to eliminate overshoot. This approach results in very slow, gentle movements, confirmed by the extremely long settling times. The goal is safety and precision over speed.

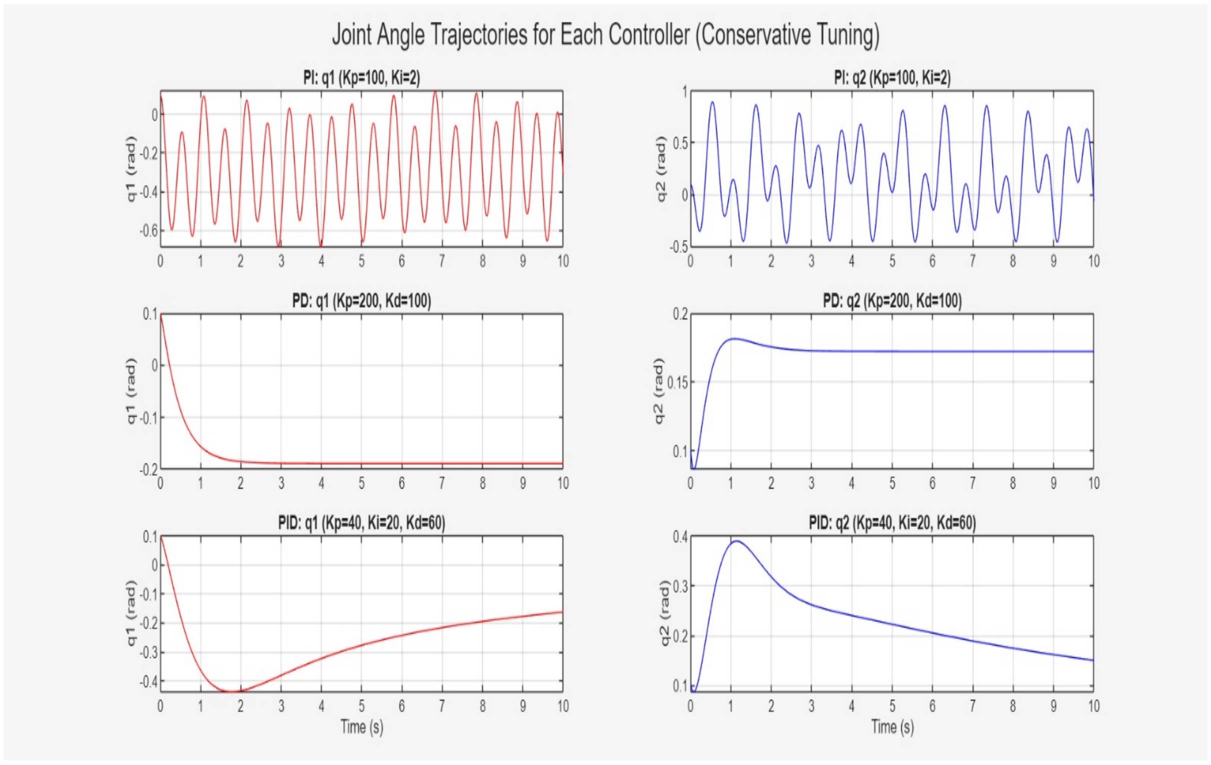


Figure 17: Conservative Tuning - Plot 1.

- **PI** ($K_p = 100, K_i = 2$)**:** Still unstable. Fails to control the massive overshoot (684.96%) even with low gains, highlighting its fundamental flaws.
- **PD** ($K_p = 200, K_d = 100$)**:** Very stable. The high K_d successfully eliminates overshoot for q2 (0.00%), creating a smooth, "overdamped" response. However, it has the largest steady-state error of any PD tuning (0.1890).
- **PID** ($K_p = 40, K_d = 60, K_i = 20$)**:** Best for Precision. The PID controller is the clear winner for this philosophy. It successfully eliminates both overshoot and steady-state error, perfectly achieving the goal of a smooth, stable, and precise motion, albeit slowly (979ms).

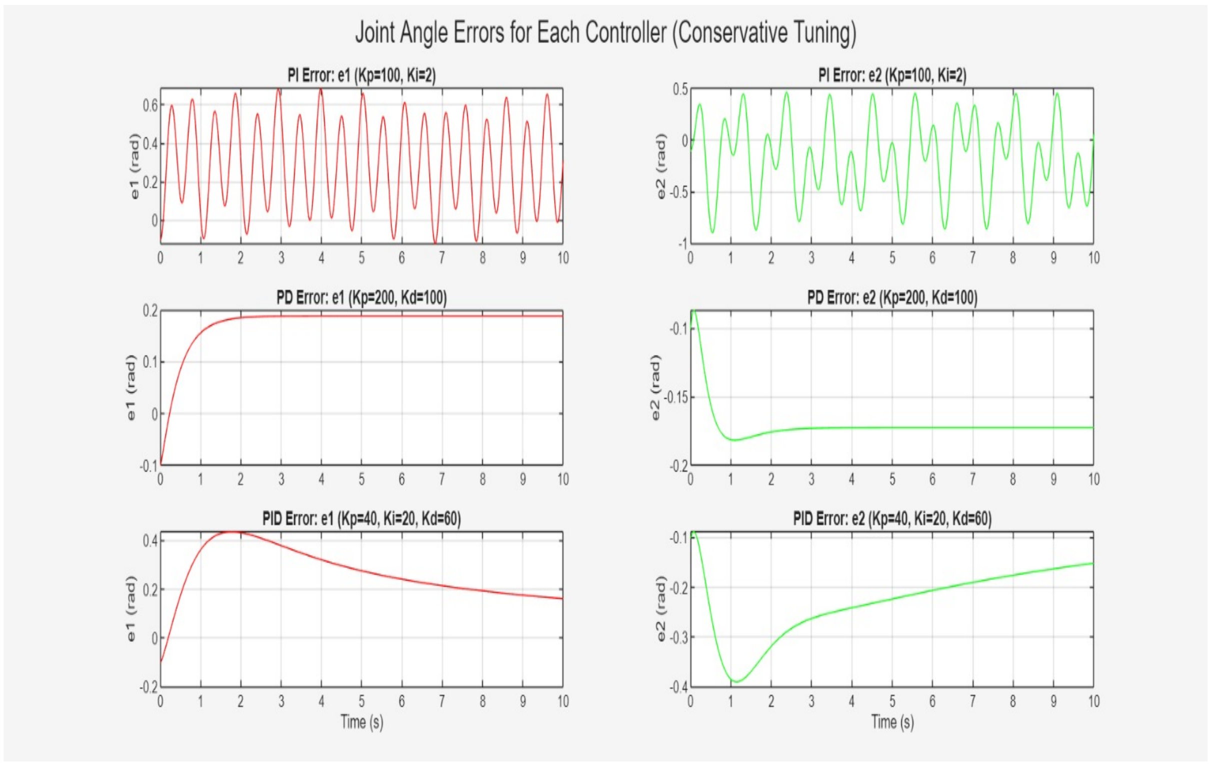


Figure 18: Conservative Tuning - Plot 2.

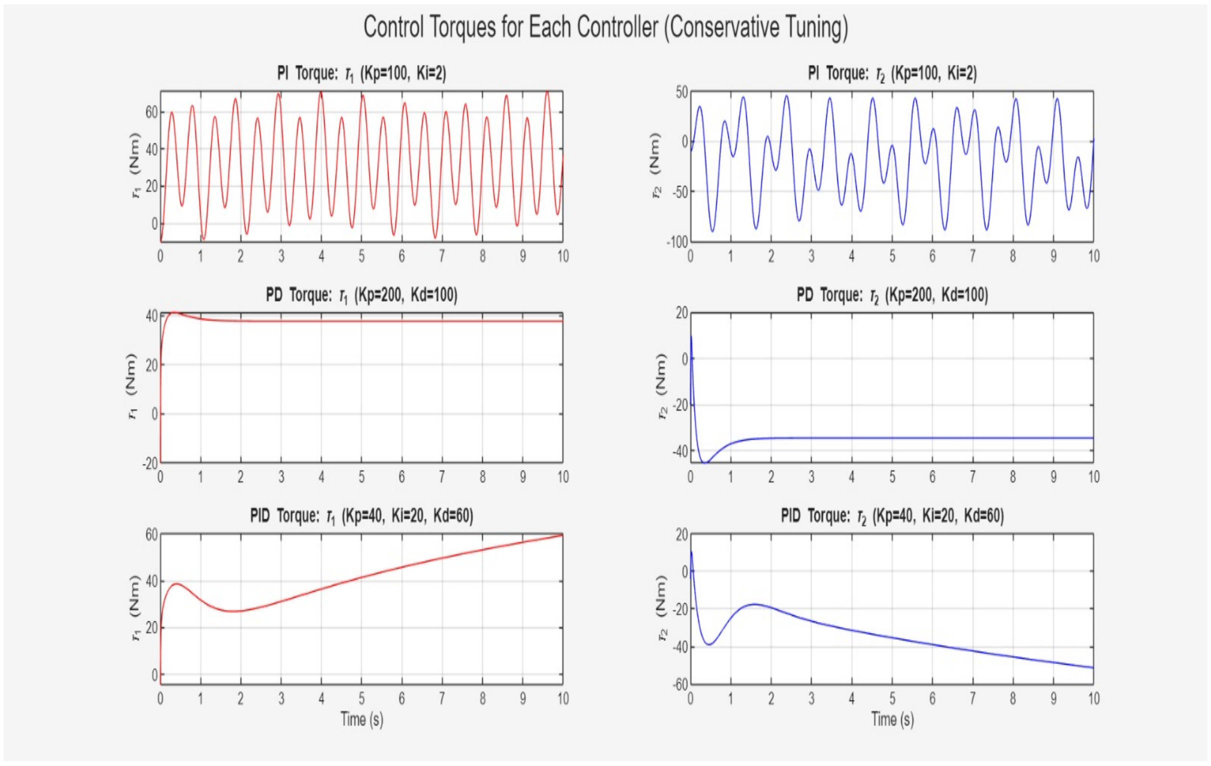


Figure 19: Conservative Tuning - Plot 3.

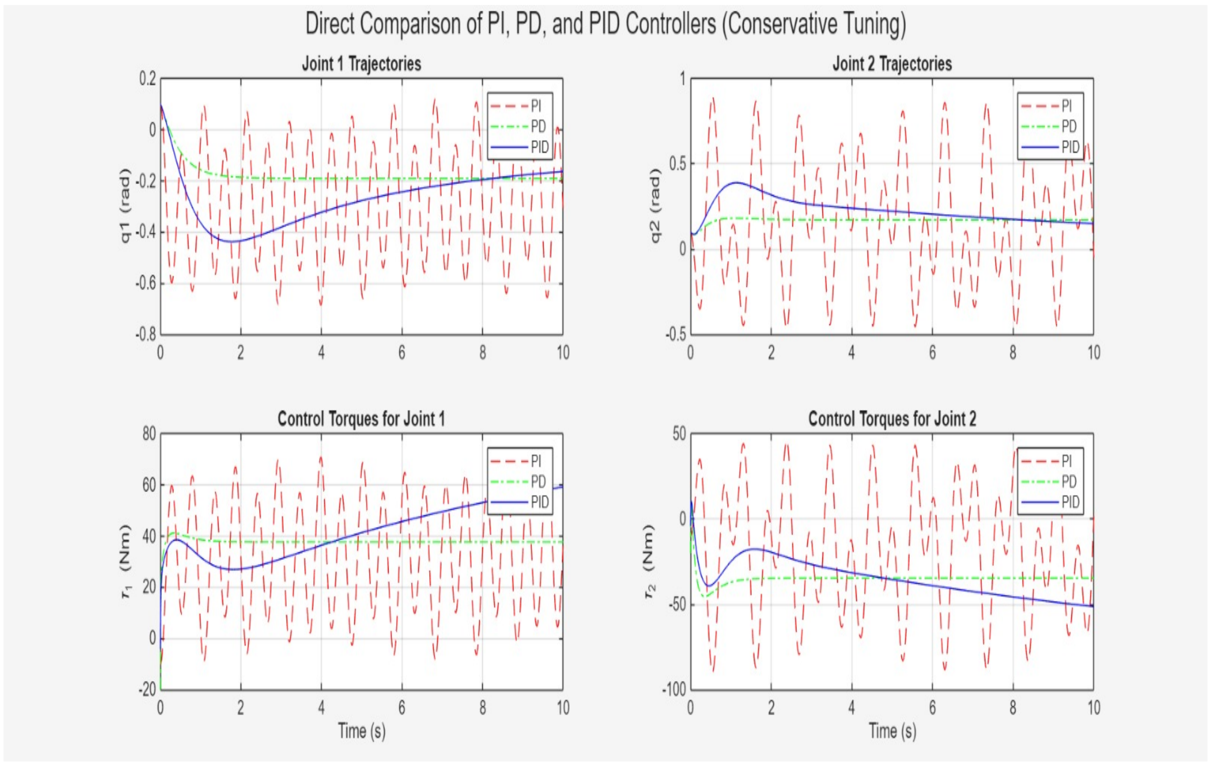
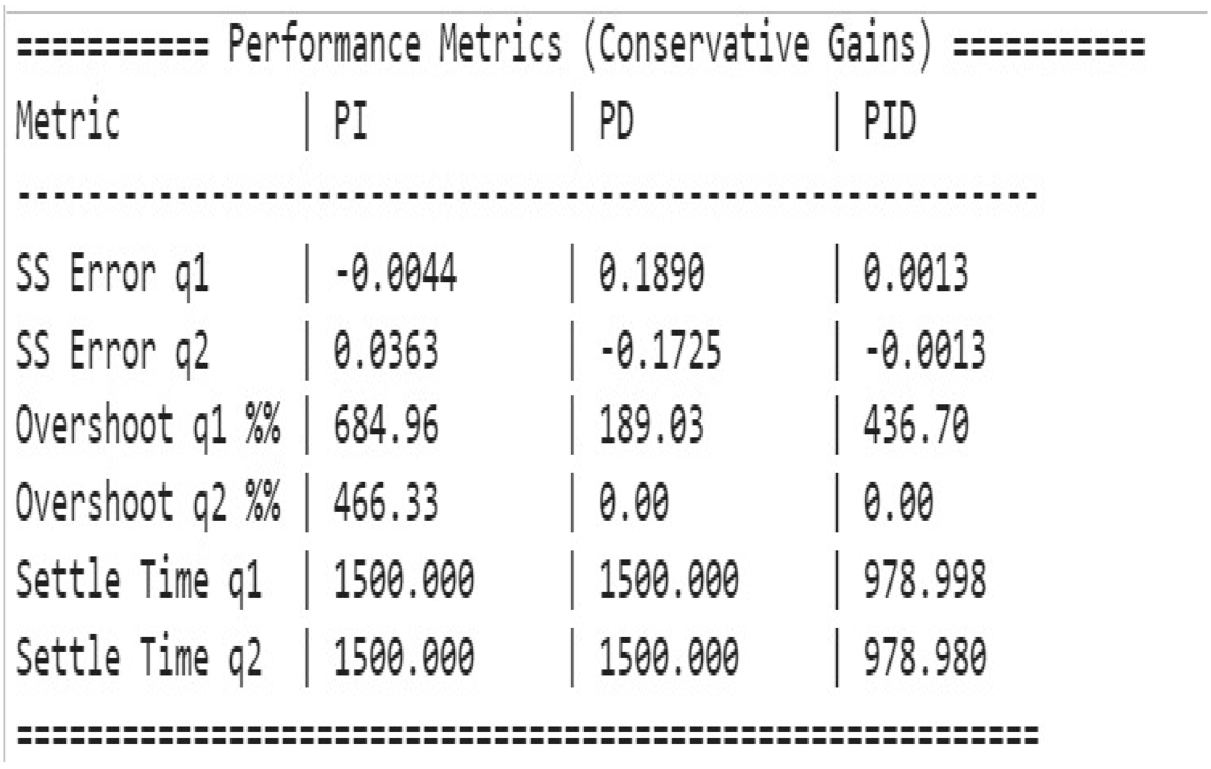


Figure 20: Conservative Tuning - Plot 4.



```
=========== Performance Metrics (Conservative Gains) ===========
Metric           | PI          | PD          | PID
-----------------------------------------------------------
SS Error q1      | -0.0044     | 0.1890      | 0.0013
SS Error q2      | 0.0363      | -0.1725     | -0.0013
Overshoot q1 %%  | 684.96      | 189.03      | 436.70
Overshoot q2 %%  | 466.33      | 0.00        | 0.00
Settle Time q1   | 1500.000    | 1500.000    | 978.998
Settle Time q2   | 1500.000    | 1500.000    | 978.980
===========================================================
```

Figure 21: Conservative Tuning - Plot 5.

7 Comprehensive Conclusion and Final Synthesis

The investigation into controlling a two-link manipulator does not yield a single "best controller" but rather a "best controller for a specific task," whose effectiveness is entirely dependent on the tuning of its gains.

- **The PI Controller** is fundamentally unsuitable for controlling this type of dynamic mechanical system. Its lack of a derivative (damping) term makes it inherently unstable, leading to severe overshoot and oscillations regardless of the tuning strategy.
- **The PD Controller** is a fast and stable option, but its performance is fundamentally limited by its inability to eliminate steady-state error. It is a good choice for applications where speed is important and a small final positioning error is acceptable.
- **The PID Controller** is the superior and most versatile solution. By combining the proportional response (K_p), derivative damping (K_d), and integral error correction (K_i), it can be precisely tailored to meet any performance objective, from aggressive speed to conservative precision.

Summary of Tuning Philosophies

Table 2: Summary of Controller Performance by Tuning Philosophy

Tuning Philosophy	Primary Goal	PD Controller Performance	PID Controller Performance	Best For...
Aggressive	Maximum Speed	Fast, but unstable with high overshoot and steady-state error.	Fastest settling time, but suffers from high overshoot.	High-speed automation where overshoot is tolerable.
Balanced	Optimal Compromise	Good speed and stability, but retains steady-state error.	Optimal. Fast, stable, and perfectly accurate.	General-purpose robotics and most applications.
Conservative	Maximum Stability	Very stable with no overshoot, but slow and has steady-state error.	Perfectly stable and accurate, but with the slowest response.	Delicate, high-precision tasks (medical, assembly).

8 Simulink

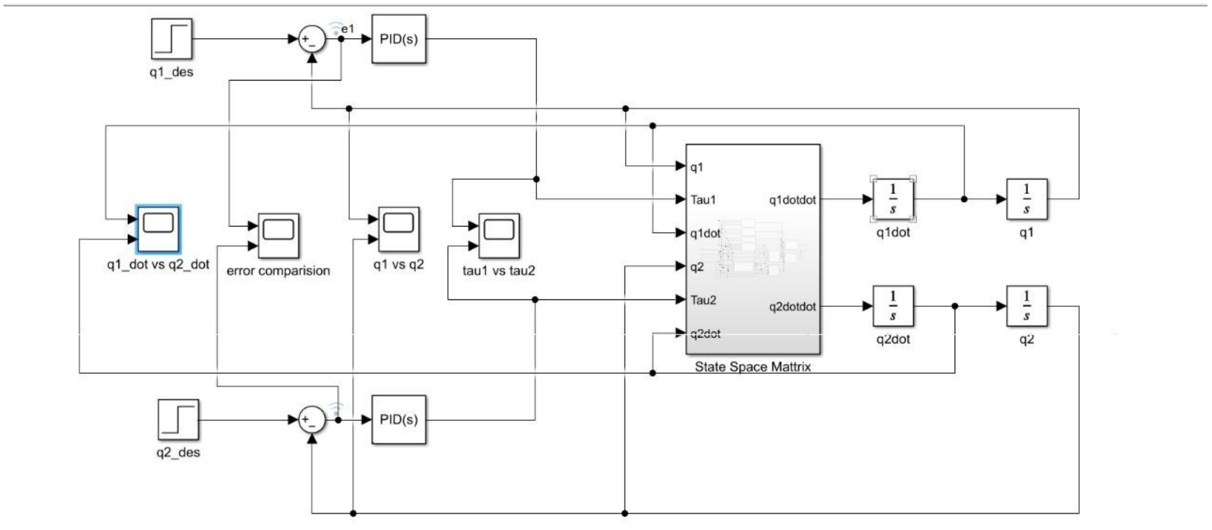


Figure 22: Closed Loop Control System

Closed-Loop Control System

1. **q1desired, q2desired:** These blocks provide the desired set points or target positions for the two links.

2. **Error Signals(e(t)):** The difference between the reference signal and the actual position (q1, q2) is used to generate an error signal, which drives the control system.

3. **Controllers (PID,PI,PD,P):** These are the feedback controllers (e.g., PID controllers) that generate control actions based on the error signals to minimize the difference between desired and actual positions.

4. **State Space:** This block represents the complete dynamics of the two-link robot, incorporating all components from the first image. It takes the control inputs and the current state to produce the system response (e.g., angular velocities and accelerations).

5. **Integration Blocks (1/s):** These blocks perform integration to convert angular accelerations (q1dotdot, q2dotdot) into angular velocities (q1dot, q2dot) and further into positions (q1, q2). This represents the dynamics of the system over time.

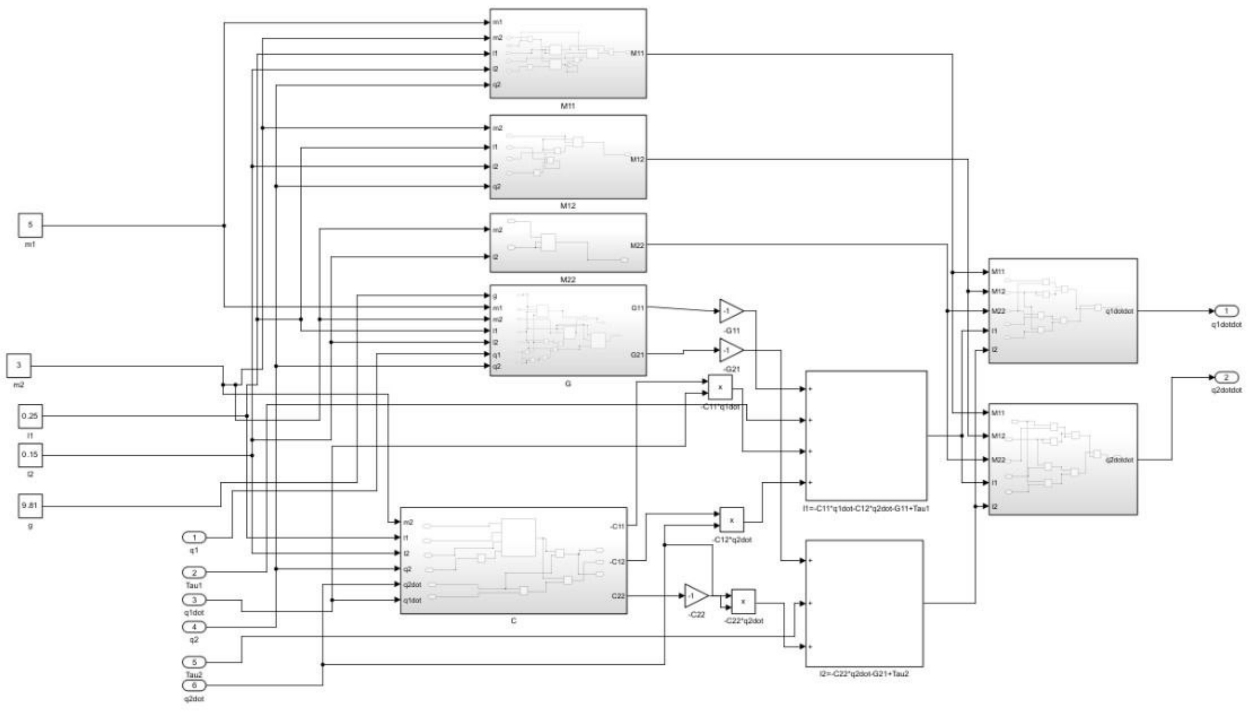


Figure 23: State Space Dynamics Model

State Space Dynamics Model

1. **M11, M12, M22:** These represent mass matrices of the two-link robot arm. They are responsible for calculating the inertia of the system. The elements of the mass matrix contribute to the dynamics of each link.

2. **G11, G21:** These are gravity vector components. They model the effect of gravitational force acting on each link based on their configuration.

3. **C11, C12, C22:** These are Coriolis and centrifugal terms, which account for the interaction between the two links due to their motion.

4. **Multipliers (x):** These are used for multiplying the Coriolis, centrifugal, and gravity terms to generate the control signals that oppose or work with these forces.

5. **Inputs m1, m2, l1, l2, etc.:** These are the physical parameters of the two-link robot such as mass (m1, m2), lengths (l1, l2), and other constants (e.g., gravitational acceleration, damping, etc.).

6. **q1, q2, q1dot, q2dot:** These are state variables representing the position (q1, q2) and velocity (q1dot, q2dot) of the two links.

7. **Outputs q1dotdot, q2dotdot:** These represent the angular accelerations of the two links, which are computed from the dynamics of the system.

Design:

The following parametric values are given: $m_1 = 5kg$, $m_2 = 3kg$, $l_1 = 0.25m$, $l_2 = 0.15m$, $g = 9.81m/s^2$. The joint angles are initially at positions $[q_1(0)\ q_2(0)] = [0.2\ 0.15]$rad.

Summary of Block diagram

1. **Set Point Generation:** The desired positions are set using q1desired and q2desired.

2. **Error Calculation:** Errors are computed as the difference between the current and target positions.

3. **Control Signal Generation:** Controllers use the errors to generate appropriate control signals for the two links.

4. **State Space Dynamics:** The state space model calculates the dynamics of the two-link robot in response to the control inputs.

5. **Integration for States:** Outputs from the state space block are integrated to update the positions and velocities.

Observation:

PID Controller

- As we observe from the previous controllers, keeping a large proportionality constant, a intermediate derivative constant to smoother the curve and a bigger integral constant to reach the steady state accurately. We can keep values as $K_p = [80, 80]$, $K_d = [25, 25]$, $K_i = [50, 50]$.

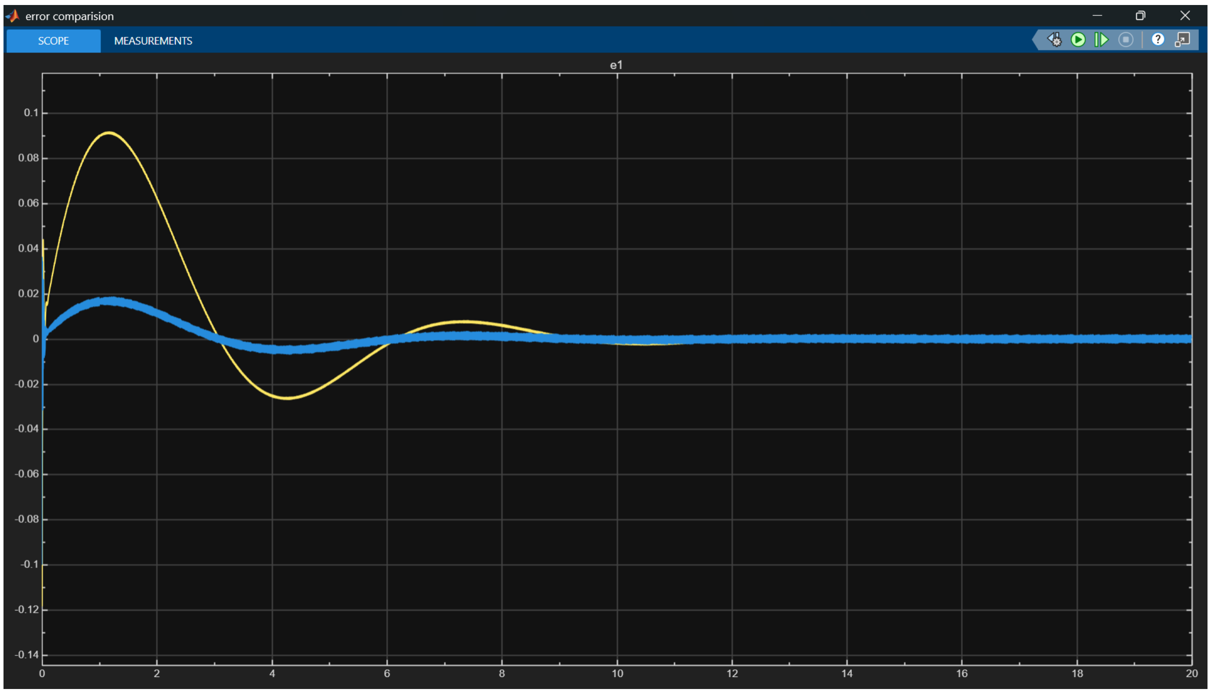


Figure 24: e1 v/s e2

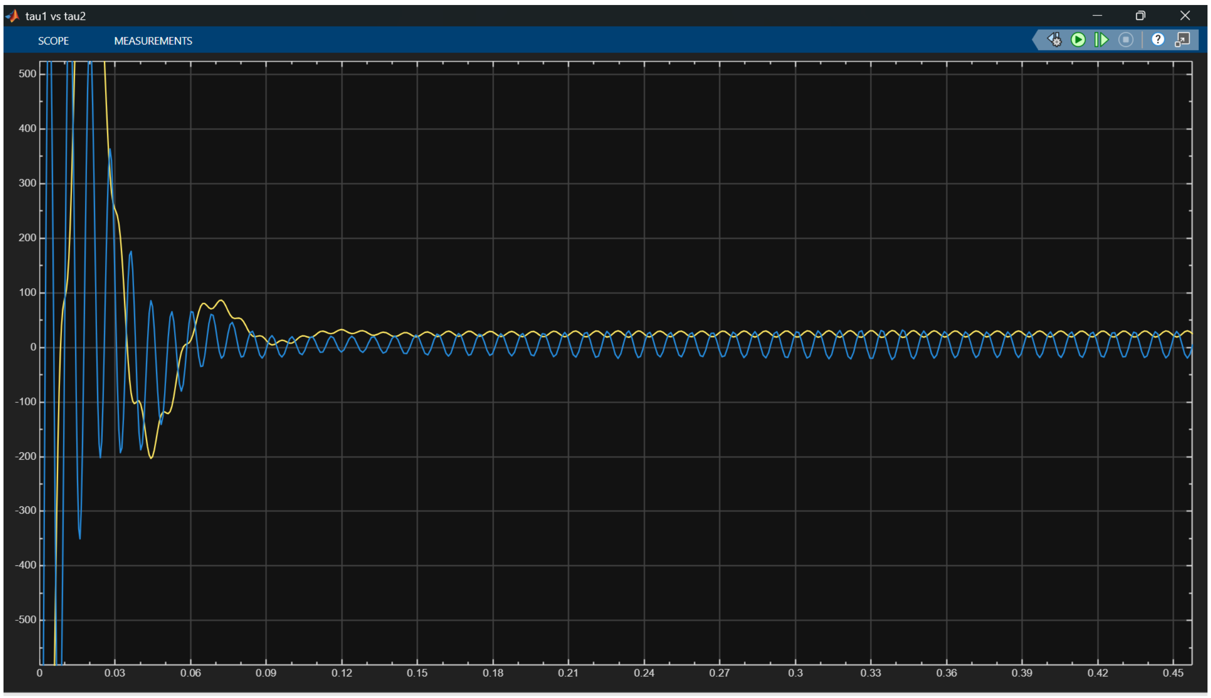


Figure 25: tau1 v/s tau2

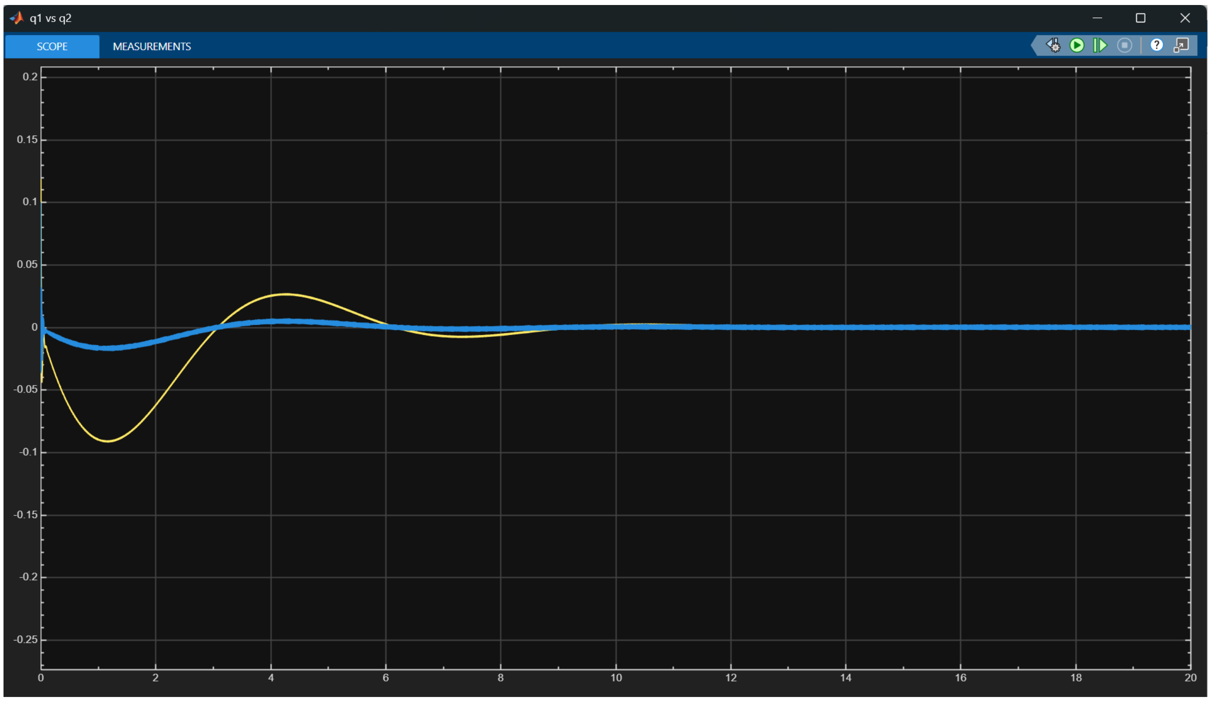


Figure 26: q1 v/s q2

As we see in figures 24, 25 and 26 we get everything we desire from the system, it reaches the steady state quickly, stabilizes faster without deviation and reaches accurately.

Team Contributions

The development of this project was a collaborative effort, with team members focusing on distinct areas of the modeling, simulation, and analysis process. The contributions are detailed below.

Controller Implementation and Performance Analysis

Team Members: Saumya, Siddhant, Madhur

This group was responsible for the implementation of the control strategies and the subsequent analysis of the simulation results. Their key contributions include:

- **Controller Implementation in MATLAB:** They developed the MATLAB scripts to implement the Proportional (P), Proportional-Derivative (PD), Proportional-Integral (PI), and Proportional-Integral-Derivative (PID) control laws. This involved scripting the logic that calculates the required torque based on system error.
- **Simulation and Tuning:** They conducted the simulations for each controller under the three distinct tuning philosophies discussed in the report: **Balanced, Aggressive, and Conservative**. This involved systematically adjusting the controller gains (K_p, K_i, K_d) to achieve the desired performance for each philosophy.
- **Analysis and Observation:** This team analyzed the data generated from the simulations to produce the plots shown in the latter half of the report, including the *Joint Angle Trajectories, Joint Angle Errors,* and *Control Torques.* They authored the written observations accompanying these plots, evaluating each controller's performance based on key metrics like settling time, overshoot, and steady-state error.

Dynamic Modeling and Simulink Implementation

Team Members: Anish, Shrenil, Het

This group focused on establishing the mathematical foundation of the 2-link manipulator and translating it into a functional simulation environment. Their contributions are:

- **Mathematical Derivation:** They derived the complete dynamic model of the 2-link manipulator from first principles using the **Lagrangian formulation**. This involved deriving the equations for the system's **Kinetic Energy (KE)** and **Potential Energy (PE)** to form the Lagrangian, and subsequently applying the Euler-Lagrange equations to obtain the final, second-order nonlinear differential equations of motion.
- **State-Space Representation:** They were responsible for converting the complex equations of motion into the standard state-space matrix form $(M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau)$, including the detailed derivation of all elements within the **Inertia** (M), **Coriolis** (C), and **Gravity** (G) matrices.

- **Simulink Model Development:** They designed and built the simulation model in **Simulink**, as shown in the report's block diagrams. This included creating the detailed *State Space Dynamics* model block that encapsulates the derived equations, as well as the top-level *Closed Loop Control System* that integrates the reference signals, error calculation, and controller blocks.