INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Systems Thinking Mini Project

Super Cool System Thinkers

Two- Link Robot Manipulator

Krrish Goenka
ID: 2023112023
krrish.goenka@research.iiit.ac.in

Prakhar Gupta
ID: 2023112019
prakhar.gup@research.iiit.ac.in

Aryan Shrivastava
ID: 2023102025
aryan.s@students.iiit.ac.in

Aman Kesharwani
ID: 2023102003
aman.kesharwani@students.iiit.ac.in

Nilavra Ghosh
ID: 2023102006
nilavra.ghosh@students.iiit.ac.in

Gautam Batra
ID: 2021112007
gautam.batra@research.iiit.ac.in

Pranav Shankar
ID: 2023112011
pranav.shankar@research.iiit.ac.in

Sajiv Singh
ID: 2023112003
sajiv.singh@research.iiit.ac.in

Abstract

Robotic manipulators are essential in manufacturing due to their speed, accuracy, and repeatability. In this paper, we present a two-link robot manipulator with rotational joints, which is modeled using Lagrange mechanics. The robot's dynamical equations are derived via the Euler-Lagrange method and then transformed into a corresponding first-order ordinary differential equations system. Then, we develop a control scheme based on a proportional-integral-derivative (PD) controller, a Proportional-Integral-Derivative controller, and a PID controller for each joint. The efficiency of these controllers was validated through simulation results. The results showed that the PID control exhibited superior performance, providing effective and precise trajectory tracking capabilities compared to the PD controller. This study provides a framework for designing and coding control algorithms for mobile robots.

Table Of Contents

1 Introduction

Robotic manipulators are essential in manufacturing due to their speed, accuracy, and repeatability. They are becoming more prevalent in daily life, with nearly every product involving their use. A proposed control model tackles the challenge of precise, fast arm movement. The double pendulum system, a nonlinear dynamic model, helps explore complex behaviors, including chaos. Using the Euler-Lagrangian equation and PID controllers, the model simulates torques applied to a two-link manipulator, mimicking human arm movements.

2 Dynamics of Two-Link Manipulators

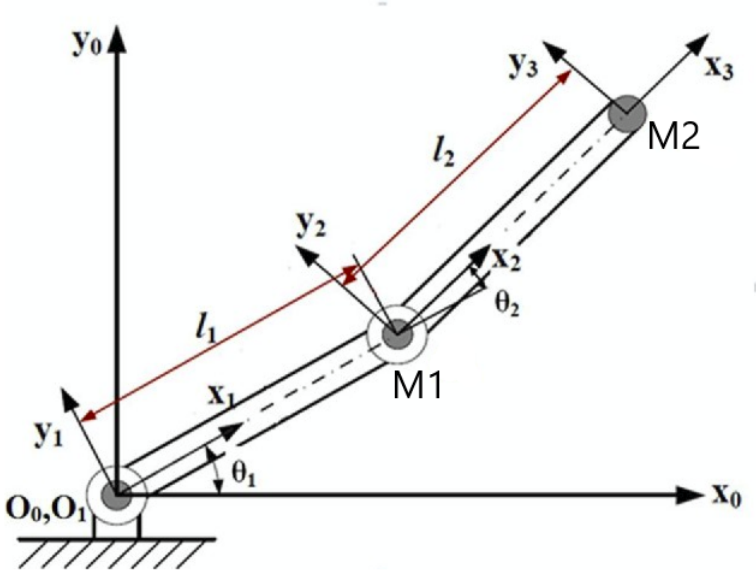

Figure 1: A simplified model of a two-link planar robot manipulator

The first step in deriving the equations of motion using the Lagrangian approach is to find the kinetic energy (KE) and the potential energy (PE) of the system.

Writing equations for x and y positions for M1:

$$x_1 = L_1 \cos(\theta_1) \tag{1}$$

$$y_1 = L_1 \sin(\theta_1) \tag{2}$$

Similarly, writing equations of position for M2:

$$x_2 = L_1 \cos(\theta_1) + L_2 \cos(\theta_2) \tag{3}$$

$$y_2 = L_1 \sin(\theta_1) + L_2 \sin(\theta_2) \tag{4}$$

We now write the velocity equations along both axes for M1:

$$v_1 = \sqrt{\dot{x}_1^2 + \dot{y}_1^2} \tag{5}$$

$$v_2 = \sqrt{\dot{x}_2^2 + \dot{y}_2^2} \tag{6}$$

Here, $\dot{x}_1$ and $\dot{x}_2$ represents the derivative of x_1 and x_2 with respect to time. In terms of θ, the variables can be written as follows:

$$\dot{x}_1 = -L_1\dot{\theta}_1 \sin(\theta_1) \tag{7}$$

$$\dot{y}_1 = L_1\dot{\theta}_1 \cos(\theta_1) \tag{8}$$

$$\dot{x}_2 = -L_1\dot{\theta}_1 \sin(\theta_1) - L_2\dot{\theta}_2 \sin(\theta_2) \tag{9}$$

$$\dot{y}_2 = L_1\dot{\theta}_1 \cos(\theta_1) + L_2\dot{\theta}_2 \cos(\theta_2) \tag{10}$$

Now, we can calculate the Kinetic Enery of both the particles separately and add them up to find the net kinetic energy(KE).

$$KE = \frac{1}{2}M_1 v_1^2 + \frac{1}{2}M_2 v_2^2. \tag{11}$$

$$\implies KE = \frac{1}{2}M_1\left(\dot{x}_1^2 + \dot{y}_1^2\right) + \frac{1}{2}M_2\left(\dot{x}_2^2 + \dot{y}_2^2\right) \tag{12}$$

Using equations 7,8,9 and 10:

$$\begin{aligned} KE = {} & \frac{1}{2}M_1\left(\left(-L_1\dot{\theta}_1\sin(\theta_1)\right)^2 + \left(L_1\dot{\theta}_1\cos(\theta_1)\right)^2\right) \\ & + \frac{1}{2}M_2\left(\left(-L_1\dot{\theta}_1\sin(\theta_1) - L_2\dot{\theta}_2\sin(\theta_2)\right)^2 + \left(L_1\dot{\theta}_1\cos(\theta_1) + L_2\dot{\theta}_2\cos(\theta_2)\right)^2\right) \end{aligned} \tag{13}$$

On simplification we get the following expression:

$$\boxed{KE = \tfrac{1}{2}(M_1 + M_2)L_1^2\dot{\theta}_1^2 + \tfrac{1}{2}M_2L_2^2\dot{\theta}_2^2 + M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)} \tag{14}$$

By definition the potential energy of the system due to gravity of the ith mass is:

$$PE_i(\theta) = M_i g h_i(\theta), \quad i = 1, 2, \tag{15}$$

where hi is the height of the center of mass of the ith body, g is the acceleration due to gravity constant, and Mi is the mass of the ith body. Therefore, the total potential energy for the system can be given by: can be given by:

$$\begin{aligned} PE = PE(\theta) &= \sum_{i=1}^{2} PE_i(\theta) \\ &= \sum_{i=1}^{2} M_i g h_i(\theta) = M_1 g L_1 \sin(\theta_1) + M_2 g\left(L_1\sin(\theta_1) + L_2\sin(\theta_2)\right) \\ \implies & \boxed{PE = (M_1 + M_2)\, g L_1 \sin(\theta_1) + M_2 g L_2 \sin(\theta_2)} \end{aligned} \tag{16}$$

Now, by Lagrangian Dynamics:

$$\mathcal{L} = KE - PE \tag{17}$$

On putting the calculated values for KE and PE from equations 14 and 16:

$$\mathcal{L} = \frac{1}{2}(M_1 + M_2)L_1^2\dot{\theta}_1^2 + \frac{1}{2}M_2L_2^2\dot{\theta}_2^2 + M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2) - (M_1 + M_2)gL_1\sin(\theta_1) - M_2gL_2\sin(\theta_2). \tag{18}$$

The Euler-Lagrange equation is given by the following equation:

$$\frac{d}{dt}\left[\frac{\partial\mathcal{L}}{\partial\dot{\theta}_i}\right] - \frac{\partial\mathcal{L}}{\partial\theta_i} = \tau_i, \quad i = 1, 2, \tag{19}$$

where τ_i is the torque applied to the ith link.

2.1 Computing Torque about both Masses

$$\frac{\partial\mathcal{L}}{\partial\dot{\theta}_i} = (M_1 + M_2)L_1^2\dot{\theta}_1 + M_2L_1L_2\dot{\theta}_2\cos(\theta_1 - \theta_2), \tag{20}$$

$$\implies \frac{\partial\mathcal{L}}{\partial\theta_1} = -M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - (M_1 + M_2)gL_1\cos(\theta_1), \tag{21}$$

Putting the above calculated result in equation 19:

$$\frac{d}{dt}\left[\frac{\partial\mathcal{L}}{\partial\dot{\theta}_1}\right] = (M_1+M_2)L_1^2\ddot{\theta}_1 + M_2L_1L_2\ddot{\theta}_2\cos(\theta_1-\theta_2) - M_2L_1L_2\dot{\theta}_2(\dot{\theta}_1-\dot{\theta}_2)\sin(\theta_1-\theta_2) \tag{22}$$

$$\frac{\partial\mathcal{L}}{\partial\dot{\theta}_2} = M_2L_2^2\dot{\theta}_2 + M_2L_1L_2\dot{\theta}_1\cos(\theta_1 - \theta_2), \tag{23}$$

$$\implies \frac{\partial\mathcal{L}}{\partial\theta_2} = M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - M_2gL_2\cos(\theta_2), \tag{24}$$

Putting the above calculated result in equation 19:

$$\frac{d}{dt}\left[\frac{\partial\mathcal{L}}{\partial\dot{\theta}_2}\right] = M_2L_2^2\ddot{\theta}_2 + M_2L_1L_2\ddot{\theta}_1\cos(\theta_1 - \theta_2) - M_2L_1L_2\dot{\theta}_1(\dot{\theta}_1 - \dot{\theta}_2)\sin(\theta_1 - \theta_2). \tag{25}$$

The expression for torque about bodies M1 and M2 can now be written as follows:

$$\boxed{\tau_1 = (M_1 + M_2)L_1^2\ddot{\theta}_1 + M_2L_1L_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) + M_2L_1L_2\dot{\theta}_2^2\sin(\theta_1 - \theta_2) + (M_1 + M_2)gL_1\cos(\theta_1)} \tag{26}$$

$$\boxed{\tau_2 = M_2L_2^2\ddot{\theta}_2 + M_2L_1L_2\ddot{\theta}_1\cos(\theta_1 - \theta_2) - M_2L_1L_2\dot{\theta}_1^2\sin(\theta_1 - \theta_2) + M_2gL_2\cos(\theta_2)} \tag{27}$$

The above equations can also be rewritten by keeping $\ddot{\theta}_1$ and $\ddot{\theta}_2$ as the subject.

$$L_1\ddot{\theta}_1 + \delta L_1L_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) = \frac{\delta\tau_1}{M_2L_1} - \delta L_1\dot{\theta}_2^2\sin(\theta_1 - \theta_2) - g\cos(\theta_1), \tag{28}$$

$$L_2\ddot{\theta}_2 + L_1\ddot{\theta}_1\cos(\theta_1 - \theta_2) = \frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^2\sin(\theta_1 - \theta_2) - g\cos(\theta_2), \tag{29}$$

where,

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

Solving for $\ddot{\theta}_1$ and $\ddot{\theta}_2$, we get the normal form of the dynamics equations:

$$\ddot{\theta}_1 = g_1\left(t, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\right), \quad \ddot{\theta}_2 = g_2\left(t, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\right), \tag{30}$$

$$g_1 = \frac{\frac{\delta\tau_1}{M_2L_1} - \delta L_2\dot{\theta}_2^2\sin(\theta_1-\theta_2) - g\cos(\theta_1) - \delta\cos(\theta_1-\theta_2)\left(\frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^2\sin(\theta_1-\theta_2) - g\cos(\theta_2)\right)}{L_1(1-\delta\cos^2(\theta_1-\theta_2))}, \tag{31}$$

$$g_2 = \frac{\frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^2\sin(\theta_1-\theta_2) - g\cos(\theta_2) - \cos(\theta_1-\theta_2)\left(\frac{\delta\tau_1}{M_2L_1} - \delta L_2\dot{\theta}_2^2\sin(\theta_1-\theta_2) - g\cos(\theta_1)\right)}{L_2\left(1-\delta\cos^2(\theta_1-\theta_2)\right)}. \tag{32}$$

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:
$u_1 = \theta_1$
$u_2 = \theta_2$
$u_3 = \dot{\theta}_1$
$u_4 = \dot{\theta}_2$
Upon differentiating, we obtain:
$\dot{u}_1 = u_3$
$\dot{u}_2 = u_4$
$\dot{u}_3 = \ddot{\theta}_1 = g_1(t, u_1, u_2, u_3, u_4)$
$\dot{u}_4 = \ddot{\theta}_2 = g_2(t, u_1, u_2, u_3, u_4)$

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

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

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

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

The initial conditions are given by:

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

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

We are taking the initial angles as:

$\theta_1 = 0.1$
$\theta_2 = 0.1$
$\dot{\theta}_1 = 0$
$\dot{\theta}_2 = 0$

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 Space Model

In control systems, a state-space model represents a system using a set of first-order differential equations. The model is defined by state variables, which describe the system's internal behavior, inputs, and outputs. It is typically expressed in matrix form: $\dot{x} = Ax + Bu$ and $y = Cx + Du$, where x represents the state vector.

The state variables are represented as:

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

This first order nonlinear differential equation of the form:
$\dot{x} = \dfrac{dx}{dt}, \quad x(0) =$ initial conditions for our state variables.

Regarding $x(0)$, considering the initial conditions as:
$x(0) = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T$

Where (a,b,c,d,e,f) are the initial values.

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 $V_{\text{error}} = V_{\text{set}} - V_{\text{sensor}}$, 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:

Term	Math Function	Effect on Control System
P	$K_p V_{\text{error}}$	Typically the main drive in a control loop, K_p reduces a large part of the overall error.
I	$K_i \int V_{\text{error}}\, dt$	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	$K_D \frac{dV_{\text{error}}}{dt}$	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.

Figure 2: PID Control Terms and Their Effects on Control Systems

From now on, θ is represented by 'q'.

$$\boxed{\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)} \tag{33}$$

where,

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

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

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

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

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

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

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

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:

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

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:

$$f_1 = K_{p1} e_1(q_1) + K_{D1} \dot{e}_1(q_1) + K_{I1} \int e_1(q_1) dt = K_{p1}(q_{1f} - q_1) - K_{D1}\dot{q}_1 + K_{I1} \int (q_{1f} - q_1) dt \tag{34}$$

$$f_2 = K_{p2} e_2(q_2) + K_{D2} \dot{e}_2(q_2) + K_{I2} \int e_2(q_2) dt = K_{p2}(q_{2f} - q_2) - K_{D2}\dot{q}_2 + K_{I2} \int (q_{2f} - q_2) dt \tag{35}$$

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

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

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

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:

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

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

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

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:

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

$$\begin{aligned} \dot{u}_1 &= \dot{q}_1 = u_3, \\ \dot{u}_2 &= \dot{q}_2 = u_4, \\ \dot{u}_3 &= \dot{q}_1 + \phi(t, u_1, u_2, u_3, u_4, u_5, u_6), \\ \dot{u}_4 &= \dot{q}_2 + \psi(t, u_1, u_2, u_3, u_4, u_5, u_6), \\ \dot{u}_5 &= \dot{x}_1 - \dot{q}_1 = -u_1, \\ \dot{u}_6 &= \dot{x}_2 - \dot{q}_2 = -u_2 \end{aligned} \tag{36}$$

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

$$\begin{bmatrix} \phi \\ \\ \psi \end{bmatrix} = -M^{-1}(q)[C(q,\dot{q})\dot{q} + G(q)] + \begin{bmatrix} K_{P1}(q_1 - u_1) - K_{D1}\dot{u}_3 + K_{I1}u_5 \\ \\ K_{P2}(q_2 - u_2) - K_{D2}\dot{u}_4 + K_{I2}u_6 \end{bmatrix} \tag{37}$$

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

5 Controller Design

5.1 P Control

When left without intervention, the two-link system exhibits chaotic behavior. In an ideal scenario, the two-link pendulum never stabilizes because internal forces drive it to an unstable state once it reaches steady state.
Employing only proportional control ensures a reduction in steady state error; however, it does not necessarily achieve the target steady state (which is 0 in this context). Proportional control also improves transient response time, making the system respond more quickly. This is because the proportional constant K_p affects the ω_n^2 term in the denominator of the standard second-order closed-loop transfer function, which then becomes:

$$G(s) = \frac{K_p\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2 K_p} \tag{38}$$

5.2 PD Control

In this case, the control strategy incorporates a derivative term based on the error in addition to the proportional gain. Thus, the controller is represented as: $K_p + K_d s$, where K_d denotes the derivative control. This is achieved by implementing the derivative block in parallel with the proportional gain block. The derivative term significantly aids in damping the transient response and mitigating its oscillatory behavior, depending on the system's inherent damping factor ζ. If ζ is particularly small, it may lead to complex poles and inevitable oscillations. However, integrating K_d helps diminish these oscillations, thereby facilitating a quicker approach to the steady state, which is largely influenced by K_p, as mentioned before. Nonetheless, it is crucial to recognize that the system may still fail to attain the desired steady state.

$$G(s) = \frac{(K_p + sK_D)\omega_n^2}{s^2 + (2\zeta\omega_n + K_D\omega_n^2)s + \omega_n^2 K_p} \tag{39}$$

5.3 PI Control

The control mechanism here incorporates both the proportional gain term and an error-integral term by integrating an integral block in parallel with the proportional gain block. Consequently, the control equation is expressed as $K_p + K_i s$. It is worth noting that this design elevates the system to third-order by introducing a third root. However, general analysis indicates that the integral term helps the transient response approach the set-point more closely, with the rate of this approach being determined by the proportional gain. Typically, the overall performance of a PI controller surpasses that of a PD controller. However, in terms of settling time, the PD controller is advantageous as it reduces oscillations and accelerates the response. The equation can be given as follows:

$$G(s) = \frac{(sK_p + K_i)\omega_n^2}{s^3 + 2\zeta\omega_n s^2 + \omega_n^2 K_p s + K_i\omega_n^2} \tag{40}$$

5.4 PID Control

As indicated by its name, this controller incorporates all the elements—proportional gain, derivative, and integral—resulting in an interaction between the constants K_p, K_d, K_i that dictates the system's behavior. This design addresses the response time (P and D), oscillations (I and D), and the achievement of the set state (P and I). In this setup, the closed-loop transfer function of the system is represented as follows -¿

$$G(s) = \frac{(K_d s^2 + sK_p + K_i)\omega_n^2}{s^3 + 2s^2(\zeta\omega_n + K_d\omega_n^2) + \omega_n^2 K_p s + K_i\omega_n^2}.$$

This controller is implemented by placing all the three control blocks in parallel with each other such that the entire setting is in series and before the open loop system transfer function.

Note: In the above equations, G(s) is the open-loop transfer function for unity feedback control system.

5.5 Control for 2 Link Manipulator

We applied PID control and adjusted the values of K_p, K_d, and K_i based on the system's response. As previously discussed, the controller parameters are applied to the system's inputs, which, in our case, is the matrix τ.

6 MATLAB

6.1 MATLAB Code

```
%Values of m,q,l,g
g=9.8;
m1=10;
m2=5;
l1=0.2;
l2=0.1;
syms q1 q2;
syms q1_dot q2_dot;
syms tau1 tau2;

q=[q1;q2];
q_dot = [q1_dot; q2_dot];
tau = [tau1; tau2];

%Defining M11, M22,M12,M21 for matrix M
M11 =(m1+m2)*(l1^2) + m2*l2*(l2+2*l1*cos(q2));
M12 =m2*l2*(l2+l1*cos(q2));
M21=M12;
M22 =m2*(l2^2);

%Defining G1 and G2 for matrix G
G1 = m1*l1*g*cos(q1)+m2*g*(l2*cos(q1+q2)+l1*cos(q1));
G2 = m2*g*l2*cos(q1+q2);

%Defining C11,C12,C21,C22
C11 = -m2*l1*l2*sin(q2)*q2_dot;
C12 = -m2*l1*l2*sin(q2)*(q1_dot+q2_dot);
C21 = 0;
C22 = m2*l1*l2*sin(q2)*q2_dot;

M = [M11, M12; M21, M22];
C = [C11, C12; C21, C22];
G = [G1; G2];
q_double_dot = (M^(-1))*(tau-C*q_dot-G);

q1_double_dot=q_double_dot(1);
q2_double_dot=q_double_dot(2);
disp(q_double_dot);
%initial values
time_vec = [0 10];
q_ini=[0.1; 0.1];
qdot_ini=[0;0];
x_ini=[0;0;];
```

```
%required values
q_req = [0; 0];

[t,states] = ode45(@compute_dynamics, time_vec,[q_ini;
   qdot_ini;x_ini]);
q1 = states(:,1);
q2=states(:,2);
q1_dot = states(:,3);
q2_dot=states(:,4);
x1=states(:,5);
x2=states(:,6);

e1= q_req(1)-q1;
e2=q_req(2)-q2;
figure
subplot (2,1,1);
plot(t,q1);
title("q_1");
xlabel("t");
ylabel("q_1(t)");
subplot (2,1,2)
plot (t,e1)
title("Error in q_1");
xlabel("t");
ylabel("e_1(t)");

figure
subplot(2,1,1)
plot(t,q2);
title("q_2");
xlabel("t");
ylabel("q_2(t)");
subplot (2,1,2)
plot (t,e2)
title("Error in q_2");
xlabel("t");
ylabel("e_2(t)");

function dxdt = compute_dynamics (t,states)
q1=states(1);
q2=states(2);
q1_dot=states(3);
q2_dot=states(4);
x1=states(5);
x2=states(6);
```

```
kp1=500;
kd1=150;
ki1=100;
kp2=500;
kd2=150;
ki2=100;
q_desired = [0;0];
e1 = q_desired(1) - q1;
e2 = q_desired(2) - q2;

f1 = kp1*e1 + ki1*x1 - kd1*q1_dot;
f2 = kp2*e2 + ki2*x2 - kd2*q2_dot;
F = [f1; f2];

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

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

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

dxdt = zeros(size(states));
dxdt(1) = q1_dot;
dxdt(2) = q2_dot;
dxdt(3) = -(5*(tau1 - (981*cos(q1 + q2))/200 - (2943*cos(q1))
   /100 + (q1_dot*q2_dot*sin(q2))/10 + (q2_dot^2*sin(q2)*(q1
   + q2))/10))/(cos(q2)^2 - 3) - (5*(2*cos(q2) + 1)*((sin(q2)
   *q2_dot^2)/10-tau2 + (981*cos(q1 + q2))/200))/(cos(q2)^2 -
    3);
dxdt(4) = (5*(2*cos(q2) + 1)*(tau1 - (981*cos(q1 + q2))/200 -
    (2943*cos(q1))/100 + (q1_dot*q2_dot*sin(q2))/10+ (q2_dot*
   sin(q2)*(q1_dot + q2_dot))/10))/(cos(q2)^2 - 3) + (5*(4*
   cos(q2) + 13)*((sin(q2)*q2_dot^2)/10- tau2 + (981*cos(q1 +
    q2))/200))/(cos(q2)^2 - 3);
dxdt(5) = q_desired(1) - q1;
dxdt(6) = q_desired(2) - q2;
end
```

This code is a MATLAB script that simulates and analyzes the control of a two-link planar robotic manipulator using a control law based on proportional-integral-derivative (PID) control.

6.2 Explanation of the Two-Link Manipulator Code

This MATLAB code models the dynamics and control of a two-link manipulator using a PID controller for each joint. Here's a breakdown of the key parts:

System Dynamics

- **Link and Mass Parameters**: m_1, m_2 are the masses of links 1 and 2, and l_1, l_2 are their lengths. g is the gravitational constant.
- **Generalized Coordinates and Inputs**:
  1. $q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$: Joint angles of links 1 and 2.
  2. $\dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$: Joint velocities.
  3. $\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$: Joint torques (control inputs).
- **Inertia Matrix (M)**: Describes how the mass is distributed, affecting the motion.

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{41}$$

  where M_{11}, M_{12}, M_{21}, and M_{22} are functions of the link lengths, masses, and joint angles.

- **Coriolis Matrix (C)**: The C matrix in the dynamics of a two-link manipulator represents the Coriolis and centrifugal forces that arise due to the motion of the links. These forces depend on the velocities of the joints and the configuration of the manipulator (joint angles).

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \tag{42}$$

- **Gravity Matrix (G)**: In the dynamics of a two-link manipulator, the G matrix represents the gravitational forces acting on the two links. This matrix accounts for how gravity affects the system's motion depending on the current configuration (joint angles).

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \tag{43}$$

- **System Equation**:

$$\ddot{q} = M^{-1}(\tau - C\dot{q} - G) \tag{44}$$

  This equation models the manipulator's motion, where joint accelerations $\ddot{q}$ are determined by torques τ, velocity effects $C\dot{q}$, and gravity G.

PID Control

- The control inputs f_1, f_2 are computed using a PID controller for each joint, trying to drive the error in joint angles e_1, e_2 to zero.

$$\boxed{\begin{aligned} f_1 &= k_{p1}e_1 + k_{i1}x_1 - k_{d1}\dot{q}_1 \\ f_2 &= k_{p2}e_2 + k_{i2}x_2 - k_{d2}\dot{q}_2 \end{aligned}} \tag{45}$$

- **Error Terms**:
  1. e_1, e_2 are the differences between the desired joint angles and the actual angles.
  2. x_1, x_2 integrate the error for the I-term of the PID.

ODE Solver

The ODE function `ode45` is used to simulate the dynamics of the system over a time range.

```
[t, states] = ode45(@compute_dynamics, time_vec, [q_ini;
   qdot_ini; x_ini]);
```

Plotting

The results for q_1, q_2 (joint angles) and their errors are plotted to observe the system's behavior over time.

```
subplot(2, 1, 1);
plot(t, q1);
subplot(2, 1, 2);
plot(t, e1);
```

Summary

This MATLAB script models the behavior of a two-link manipulator controlled by a PID algorithm. It starts by setting parameters for masses, lengths, and gravity, and proceeds to calculate the mass matrix $\mathbf{M}$, Coriolis matrix $\mathbf{C}$, and gravitational vector $\mathbf{G}$ from joint positions. Joint accelerations are obtained by solving the system's dynamic equations. The ODE45 solver is used to simulate the temporal changes in joint angles, velocities, and errors, with PID torques directing the movement.

6.3 Simulations and Results

6.3.1 PD CONTROLLER

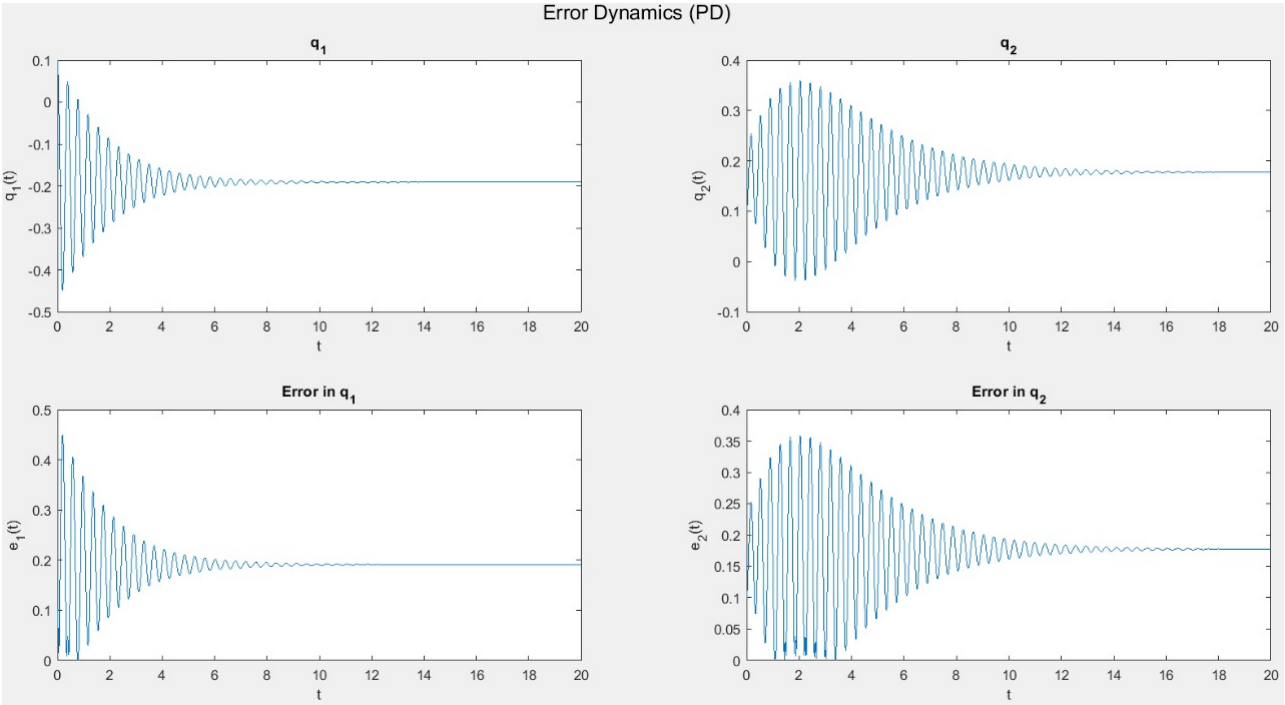

Figure 3: $K_{p1} = 250$, $K_{p2} = 250$, $K_{d1} = 1$, $K_{d2} = 1$

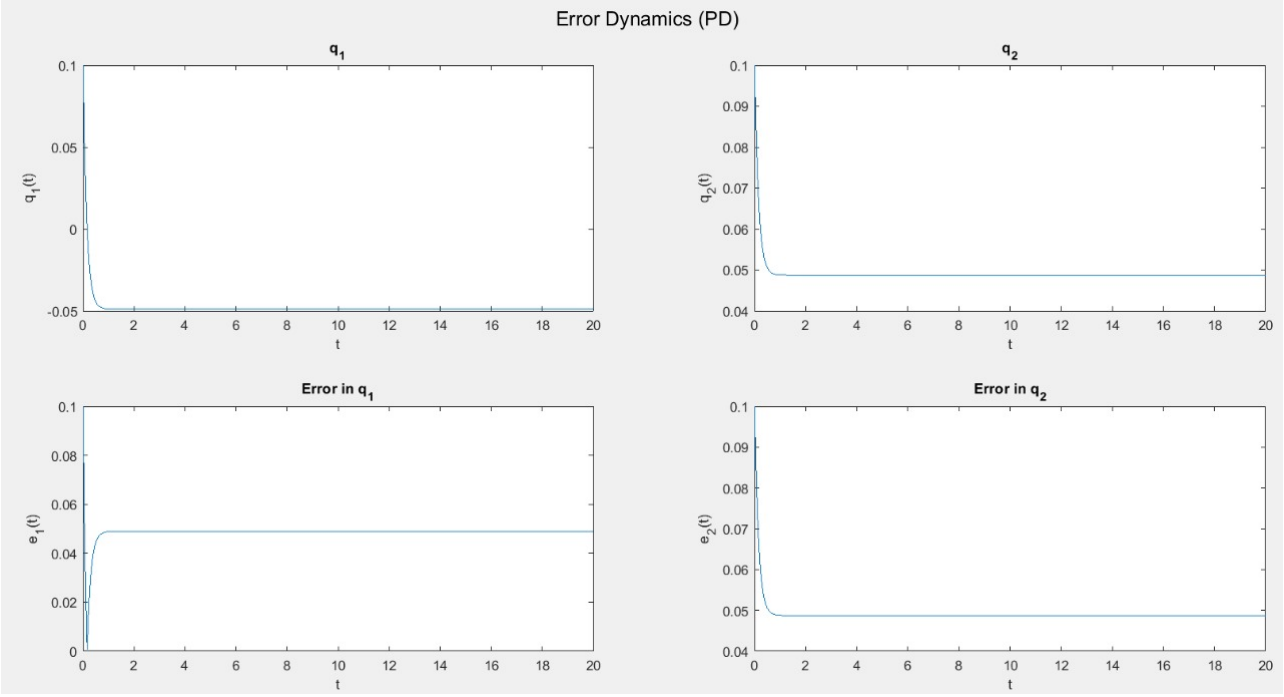

Figure 4: $K_{p1} = 1000$, $K_{p2} = 1000$, $K_{d1} = 150$, $K_{d2} = 150$

6.3.2 PI CONTROLLER

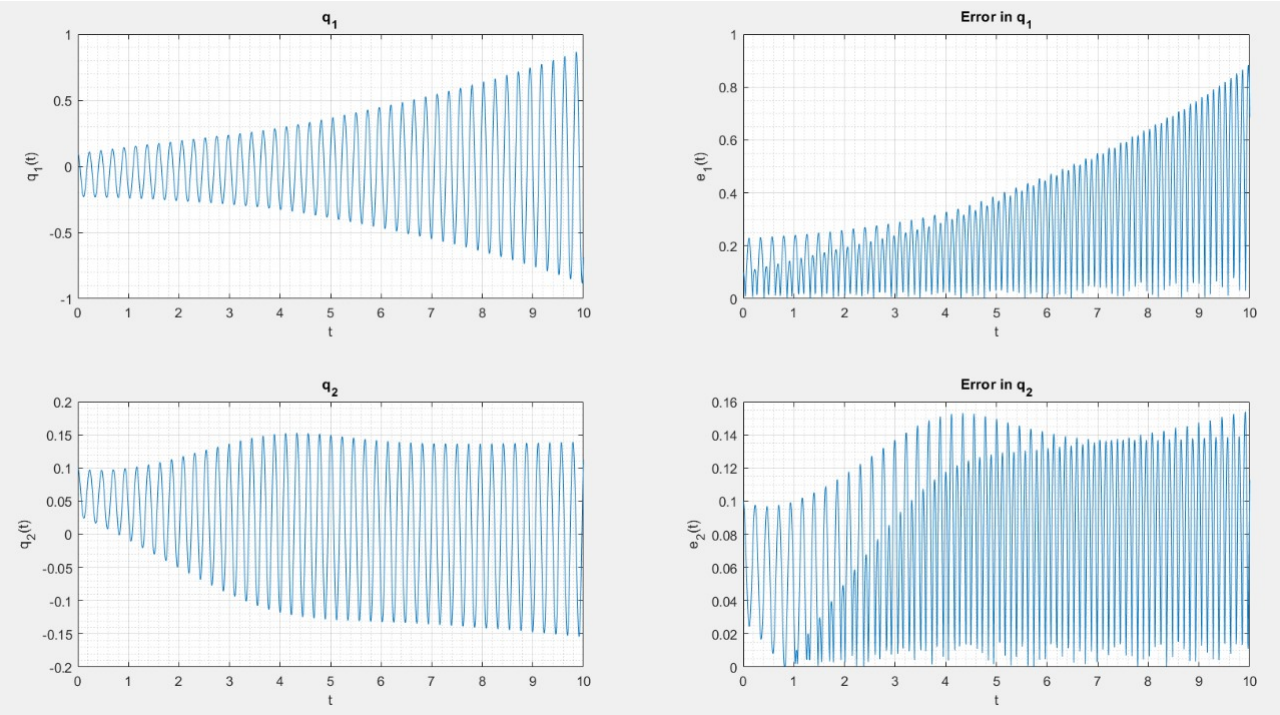

Figure 5: $K_{p1} = 750$, $K_{p2} = 750$, $K_{i1} = 250$, $K_{i2} = 250$

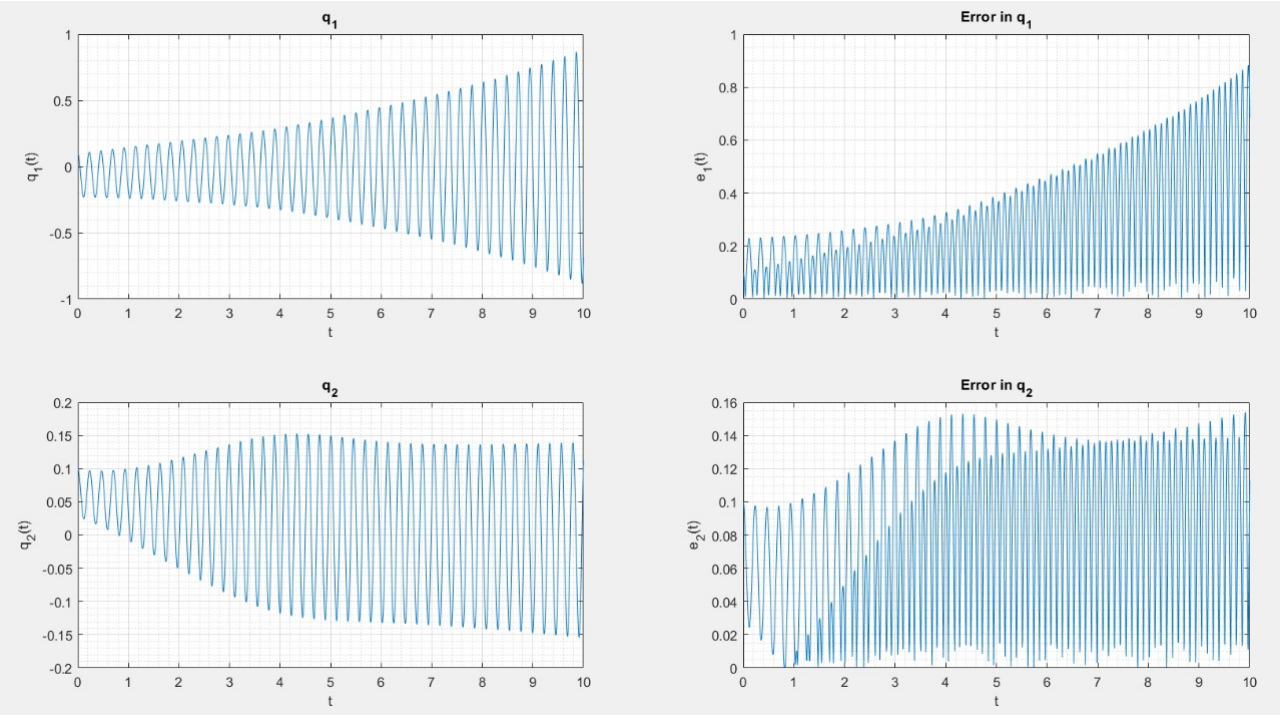

Figure 6: $K_{p1} = 1500$, $K_{p2} = 1500$, $K_{i1} = 200$, $K_{i2} = 200$

6.3.3 PID CONTROLLER

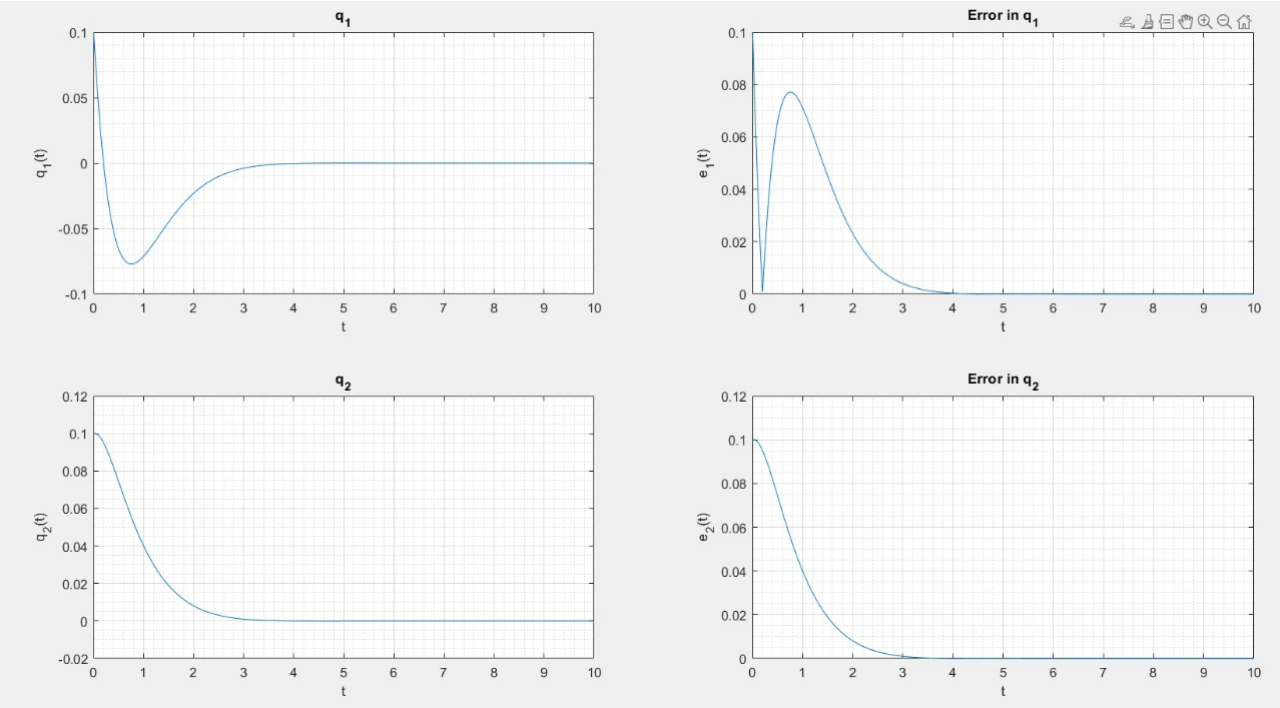

Figure 7: $K_{p1} = 500,\ K_{p2} = 500,\ K_{d1} = 150,\ K_{d2} = 150,\ K_{i1} = 500,\ K_{i2} = 500$

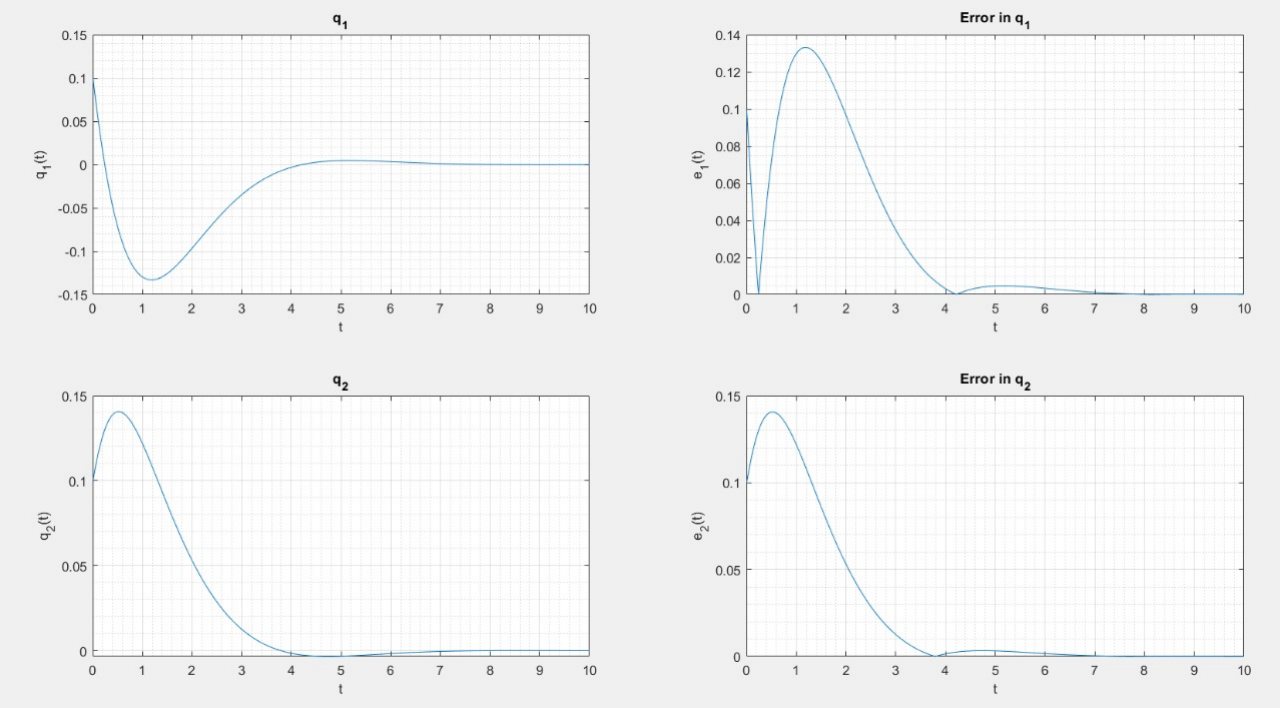

Figure 8: $K_{p1} = 250,\ K_{p2} = 250,\ K_{d1} = 250,\ K_{d2} = 250,\ K_{i1} = 200,\ K_{i2} = 200$

7 Observations and Results

From the above simulations and results we observe that the two-link manipulator's controller is adjusted to fine-tune the values of Kp (proportional gain), Kd (derivative gain), and Ki (integral gain) for optimal performance regarding stability, steady-state error, transient response, and system responsiveness.
The control system must be stable, minimizing errors. It should avoid sustained oscillations or uncontrolled divergence. Transient response characteristics include rise time, settling time, and overshoot. Reducing overshoot improves accuracy but may cause sluggishness if Kp is significantly lowered. Less overshoot means the system smoothly approaches the desired setpoint without significantly exceeding it. Higher overshoot results in a quicker response. The steady-state error must be minimized to maintain accurate setpoints.

Role of K_p: Very high Kp values can cause instability, leading to oscillations or divergence. P-only control reduces fluctuations in the process variable but may not always reach the setpoint. It provides a quicker initial response compared to other controllers, potentially reacting a few seconds faster. As system complexity increases, the response time advantage might accumulate, allowing the P-controller to respond even minutes faster. Despite the faster response, P-only control introduces an offset, which is generally undesirable. Reducing Kp makes control less aggressive, leading to less overshoot but may cause a slow response if reduced too much.

Role of K_i: Integral control (Ki) eliminates steady-state error by accelerating error correction and reducing deviations over time. It removes offset but can destabilize the controller and contribute to overshoot if set too high. Ki adjustments should be incremental to avoid excessive overshoot.

Role of K_d: Derivative control (Kd) dampens oscillations and predicts error rate changes. Increasing Kd helps stabilize the system and reduces overshoot. It shortens settling time by minimizing oscillations, allowing the system to reach the setpoint quickly without excessive oscillatory behavior.

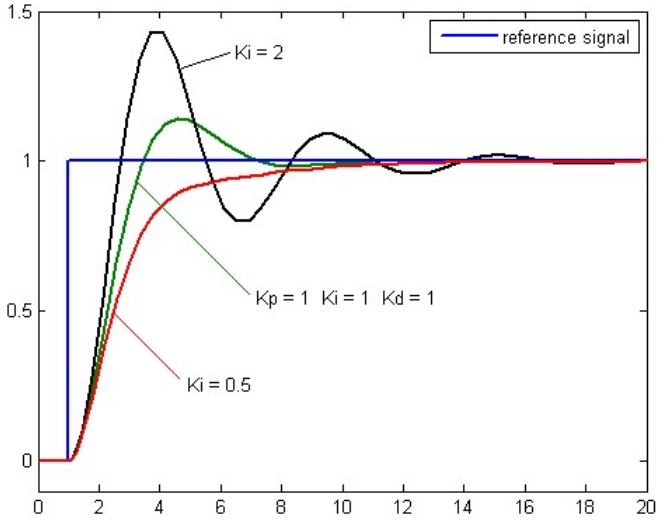

Figure 9: System Designed in Simulink

8 Simulink

8.1 System Design

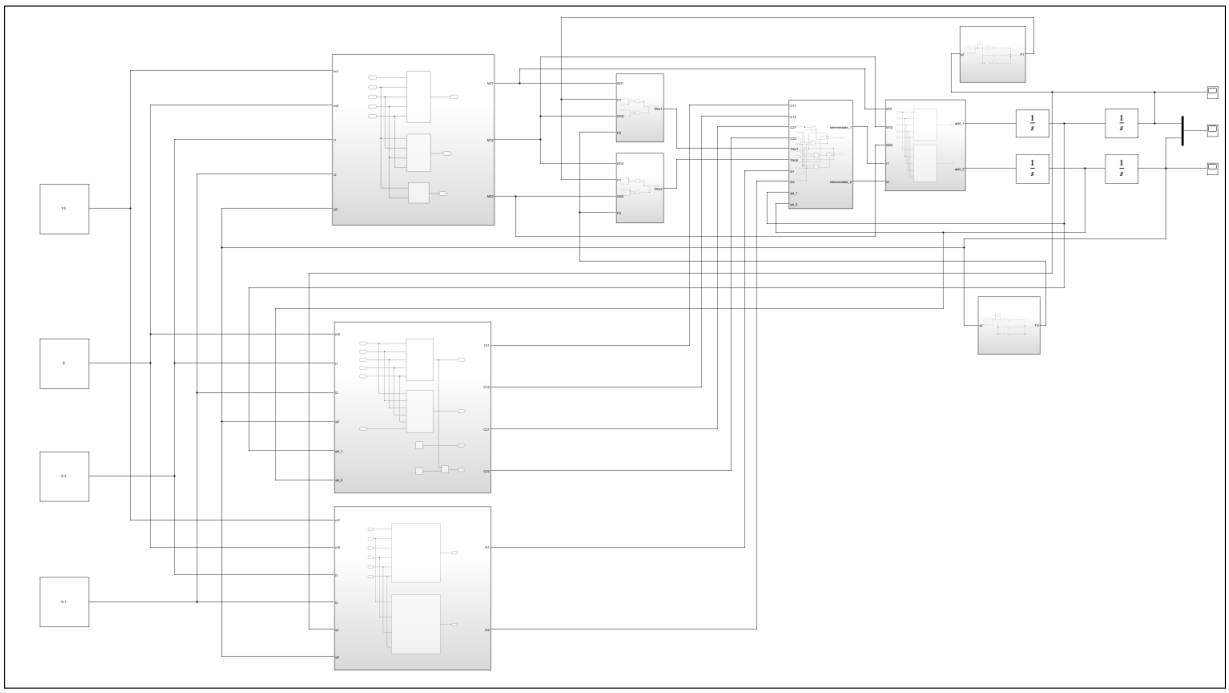

Figure 10: System Designed in Simulink

8.2 Simulation Results

8.2.1 PD Controller

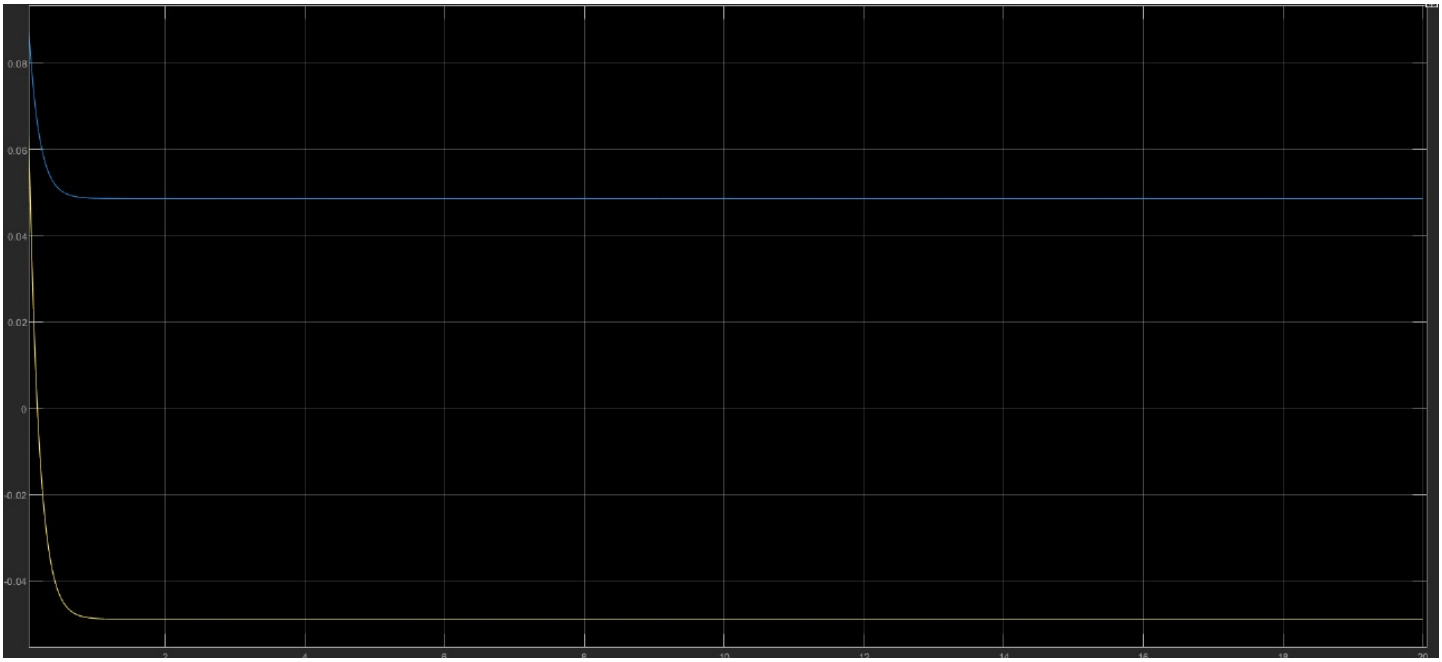

Figure 11: Kp=150, Kd=100

8.2.2 PI Controller

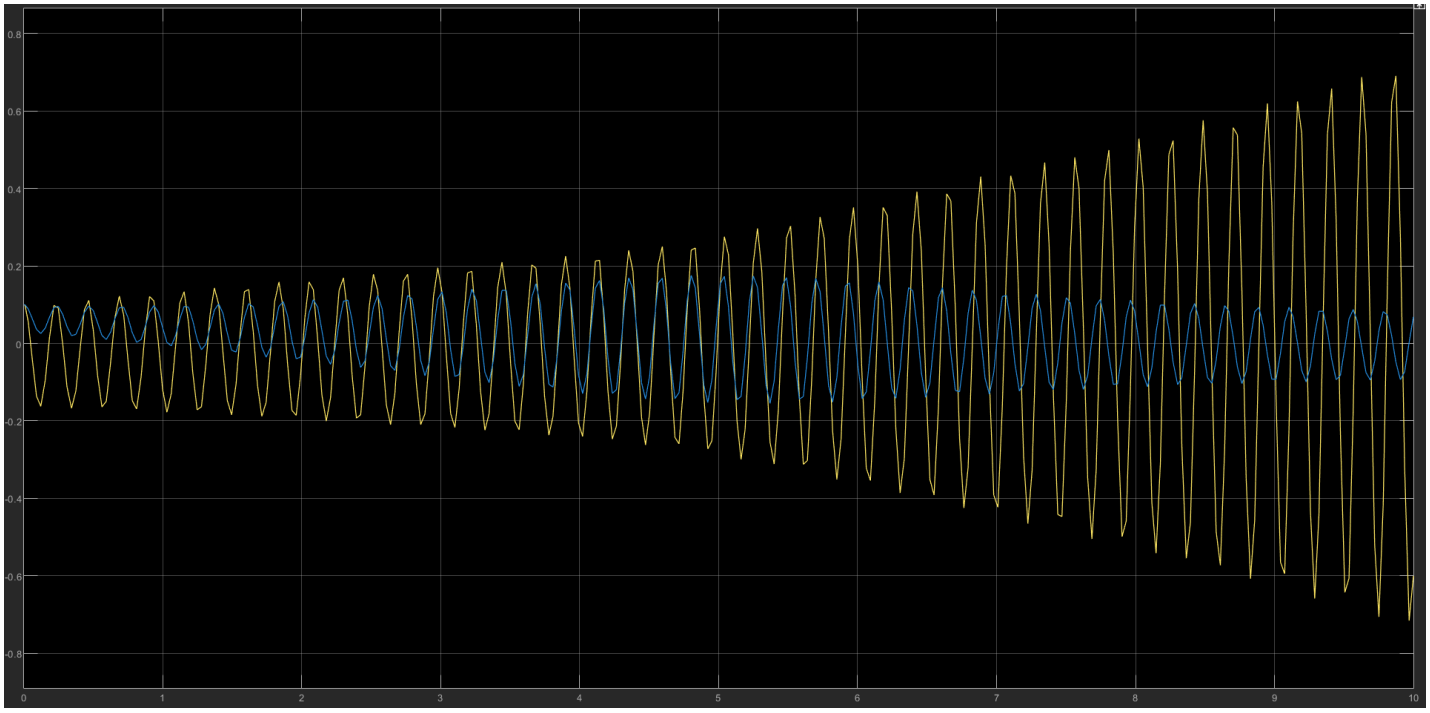

Figure 12: Kp=750, Ki=250

8.2.3 PID Controller

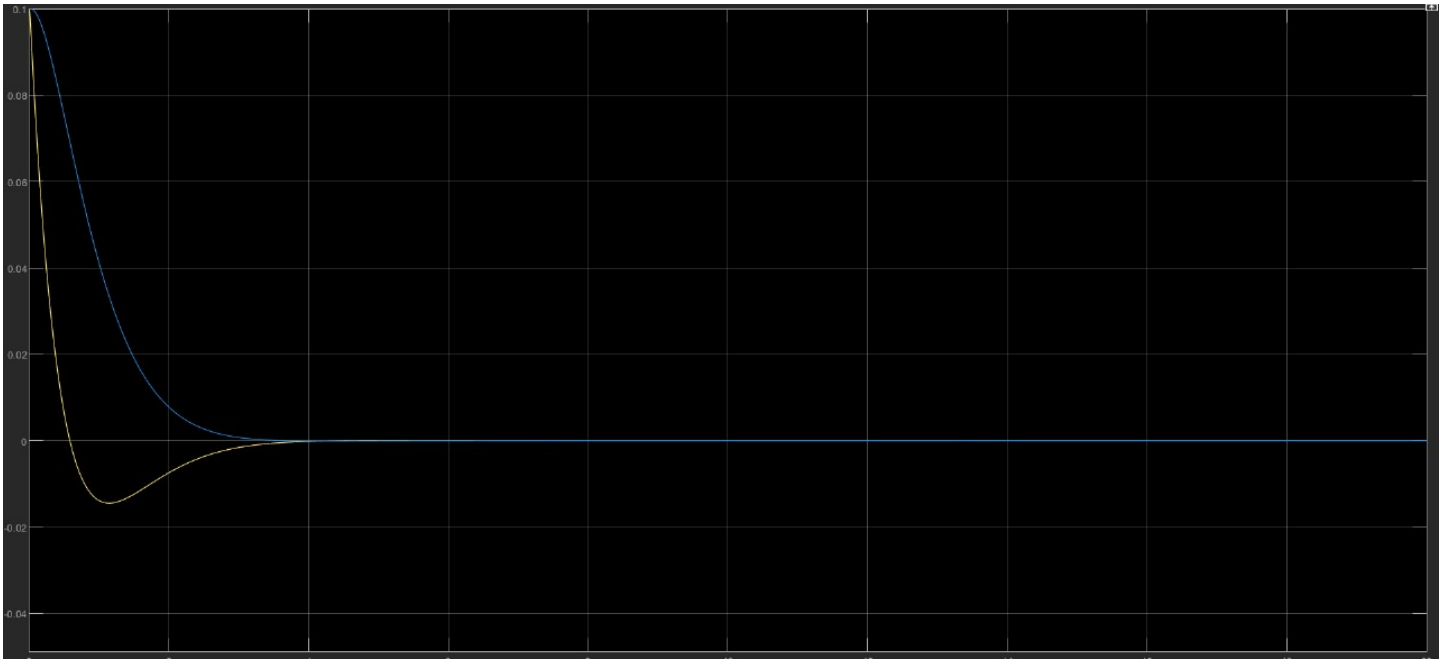

Figure 13: $K_p = 500$, $K_i = 500$ and K_d =150

9 Conclusion

This report focuses on a two-link robotic manipulator, with its dynamics modeled using Lagrange mechanics. Initially, we derived the dynamical equation of the two-link robot manipulators via the Euler-Lagrange method and then developed a straightforward and efficient control scheme. Specifically, we introduced a Proportional controller, a Proportional-Derivative controller, and a Proportional-Integral-Derivative (PID) controller to manage the robot's motion to a desired position. We detailed how PD, PI, and PID controllers can maintain the links at a required position. The efficiency of these controllers was validated through simulation results. This model offers a framework for designing and coding control algorithms for mobile robots. In this project, we examined a planar robot with rotational joints. The performance of the two-link robotic manipulator was assessed using PD, PI, and PID control. The PID controller exhibited superior performance, providing effective and precise trajectory tracking capabilities compared to the PD controller. Moreover, the PID controller demonstrated reduced oscillations around the desired trajectory in comparison to PD and PI controllers.