System Thinking Project

2-Link Manipulator

Team : *Kalman Knights*

Aarav Singla - 2024122004

Aasrith Reddy Vedanaparti - 2023102031

Aditya Peketi - 2024122001

Akshat Puneet - 2024122005

Hrishikesh Milind Gawas - 2024122006

Phanindra Manoj Kumar Kotha - 2023102008

Soham Jahagirdar - 2023102046

Somish Singh Nol - 2024122003

Nagalla Dinesh - 2023102002

International Institute of Information Technology,Hyderabad

Contents

Abstract

In this paper we propose and analyse a proportional integral-derivative (PID) controller for a **classic 2-link robotic manipulator**. First we put forth equations of motion of the two link robotic manipulator. We focus mainly on control of the robot manipulator to get the desired position using the computed torque control method. Control simulations is performed using MATLAB. Several computer simulations are used to verify the performance of the controller. Specifically we are interested in simulating the system performance under proportional-derivative (PD), proportional-integral (PI) and proportional-integral-derivative (PID) control for differing values of the respective control gains (k_p, k_i, k_d). Our findings and analysis is also verified using simulations run on MATLAB's SIMULINK.

1 Dynamics of Two-Link Manipulator

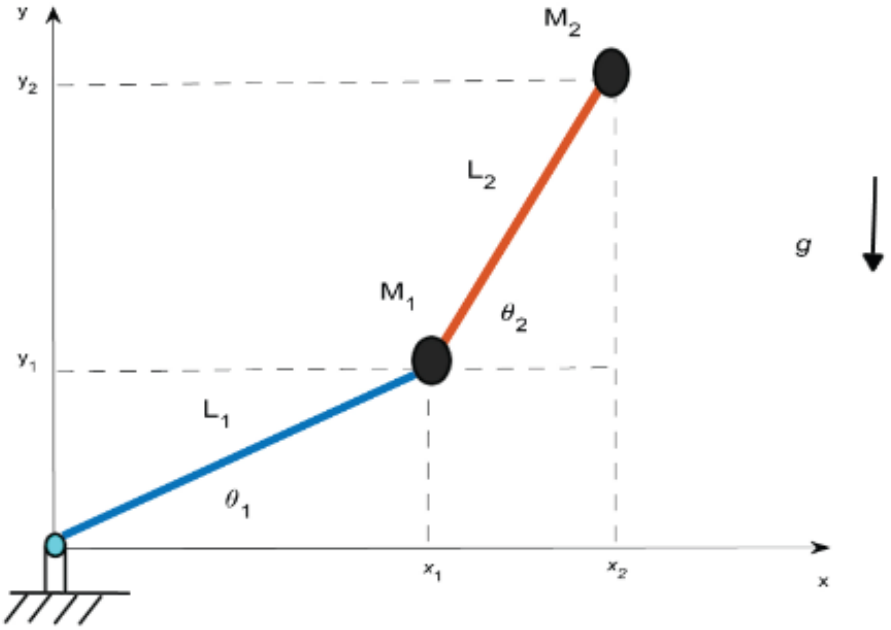


Figure 1: The Two-Link Manipulator

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 :

$$x_1 = L_1 cos(\theta_1)$$

$$y_1 = L_1 sin(\theta_1)$$

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

$$x_2 = L_1 cos(\theta_1) + L_2 cos(\theta_1 + \theta_2)$$

$$y_1 = L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2)$$

We know that

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

$$r = xi + yj$$

$$\frac{dr}{dt} = \frac{dx}{dt}i + \frac{dy}{dt}j$$

$$v = \frac{dx}{dt}i + \frac{dy}{dt}j$$

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 :

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

Energy information of the System.

The general formula for **Kinetic Energy** (KE) is

$$KE = \frac{1}{2}mv^2$$

Therefore Kinetic Energy of above System is given by

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

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

$$KE = \frac{1}{2}M_1L_1^2\dot{\theta}_1^2 + \frac{1}{2}M_2\left(L_1^2 + L_2^2\right)\dot{\theta}_1^2 + M_2L_2^2\dot{\theta}_1\dot{\theta}_2cos(\theta_1 - \theta_2) + \frac{1}{2}M_2L_2^2\dot{\theta}_2^2$$

The general formula for **Potential Energy** (PE) is

$$PE = Mgh$$

Here h is the height at which mass M is located from ground.
In our system, Mass M_1 is located at height $L_1\sin(\theta_1)$ and Mass M_1 is located at height $L_1\sin(\theta_1) + L_2\sin(\theta_2 + \theta_2)$.
Therefore Potential Energy of above System is given by

$$PE = M_1gL_1sin(\theta_1) + M_2g(L_1sin(\theta_1) + L_2sin(\theta_2 + \theta_2))$$

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 $q_1, q_2, q_3, q_4...$ 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 is

$$\frac{d}{dt}(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}) - \frac{\partial \mathcal{L}}{\partial q_i} = 0$$

For our System we get :

The Lagrangian is given by:

$$\mathcal{L} = \frac{1}{2}(m_1+m_2)l_1^2\dot{\theta}_1^2+\frac{1}{2}m_2l_2^2\dot{\theta}_2^2+m_2l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2)-(m_1+m_2)gl_1\sin(\theta_1)-m_2gl_2\sin(\theta_2)$$

We select general coordinates θ_1 and θ_2 .
So,The Euler-Lagrange equation is given by the equation:
Therefore, the following are two nonlinear equations of motion which are second-order system of ordinary differential equations:

$$l_1\ddot{\theta}_1 + \delta l_1 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) = \frac{\delta\tau_1}{m_2 l_1} - \delta l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - g\cos(\theta_1)$$

$$l_1\dot{\theta}_2 + l_1\ddot{\theta}_1 \cos(\theta_1 - \theta_2) = \frac{\tau_2}{m_2 l_2} + l_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - g\cos(\theta_2)$$

where

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

From the above equations, we get the following values for $\ddot{\theta}_1$ and $\ddot{\theta}_2$:

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

where

$$g_1 = \frac{\frac{\delta\tau_1}{m_2 l_1} - \delta l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - g\cos(\theta_1) - \delta\cos(\theta_1 - \theta_2)\left(\frac{\tau_2}{m_2 l_2} + l_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - g\cos(\theta_2)\right)}{l_1(1 - \delta\cos^2(\theta_1 - \theta_2))}$$

$$g_2 = \frac{\frac{\tau_2}{m_2 l_2} + l_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - g\cos(\theta_2) - \cos(\theta_1 - \theta_2)\left(\frac{\delta\tau_1}{m_2 l_1} - \delta l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - g\cos(\theta_1)\right)}{l_2(1 - \delta\cos^2(\theta_1 - \theta_2))}$$

In order to solve for the angles θ_1 and θ_2, let us consider four variables:

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

After differentiating with respect to time:

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

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

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

where

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

with

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

The initial conditions are given by:

$$U_0 = \begin{bmatrix} u_1(0) \\ u_2(0) \\ u_3(0) \\ u_4(0) \end{bmatrix}$$

where

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

2 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's behavior 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:

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

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

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

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** $C(q, \dot{q})$

- $C_{11} = -m_2 l_1 l_2 sin(q_2)\dot{q}_2$
- $C_{12} = -m_2 l_2 l_1 sin(q_2)(\dot{q}_1 + \dot{q}_2)$

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

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 $m_1 = 10\,\text{kg}$, length $l_1 = 0.2\,\text{m}$
- Link 2: mass $m_2 = 5\,\text{kg}$, length $l_2 = 0.1\,\text{m}$

The gravitational acceleration is $g = 9.81\,\text{m/s}^2$.
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

$$\ddot{q} = -M(q)^{-1}C(q,\dot{q})\dot{q} - M(q)^{-1}G(q) + M(q)^{-1}\tau \tag{2}$$

$$\ddot{q} = -M^{-1}C\dot{q} - M^{-1}G + M^{-1}\tau \tag{3}$$

2.1 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:

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

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

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

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

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

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

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:

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

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.

3 Control Systems

Controllers are critical components in a 2-link manipulator system due to the following reasons:

1. **Precision and Accuracy:** Controllers help achieve precise positioning and orientation of the manipulator's end effector, which is essential for tasks like assembly, welding, or painting.

2. **Trajectory Tracking:** They allow the manipulator to follow a desired path or trajectory smoothly and accurately, ensuring that the end effector moves along predefined routes without deviations.

3. **Dynamic Response:** Controllers enable the manipulator to respond dynamically to changes in the environment, such as variations in load or unexpected obstacles, maintaining stability and performance.

4. **Error Correction:** Through feedback mechanisms, controllers can detect errors in positioning and make necessary adjustments in real-time, enhancing overall accuracy.

5. **Force Control:** Controllers manage the forces exerted by the end effector, ensuring safe interaction with objects, preventing damage, and allowing for compliant behavior when needed.

6. **Complexity Reduction:** They simplify the control of complex manipulator dynamics by breaking down tasks into manageable components, making control system design more straightforward.

7. **Simulation and Optimization:** Controllers facilitate the simulation of manipulator dynamics, allowing for testing and optimization of control strategies before physical implementation.

Three types of control actions :

- Proportional (P)
- Integral (I)
- Derivative (D)

Here's how each component effects the performance of a 2-link manipulator:
Effect:

- **Proportional Control (P)** The proportional component produces an output that is proportional to the current error (the difference between the desired position and the actual position).
- **Integral Control (I)** The integral component accumulates the error over time, integrating past errors to eliminate steady-state error.
- **Derivative Control (D)** The derivative component predicts future errors based on the rate of change of the error, providing a damping effect on the system.

3.1 Perquisites

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

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

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

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

3.2 Proportional Derivative Controller

A common technique for controlling a system with input is to use the following general structure of PD controller with error e

$$f = K_p e + K_d \frac{d(e)}{dt}$$

By applying Laplace Transformation on above equation

$$F(s) = K_p E(s) + K_d E(s).s$$

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

$$f_1 = K_p e_1 + K_d \frac{d(e_1)}{dt}$$

$$f_1 = K_p(q_{1f} - q_1) + K_d \frac{d(q_{1f} - q_1)}{dt}$$

$$f_2 = K_p e_2 + K_d \frac{d(e_2)}{dt}$$

$$f_2 = K_p(q_{2f} - q_2) + K_d \frac{d(q_{2f} - q_2)}{dt}$$

3.3 Proportional Integral Controller

A common technique for controlling a system with input is to use the following general structure of PI controller with error e

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

By applying Laplace Transformation on above equation

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

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

$$f_1 = K_p e_1 + K_i \int e_1 dt$$

$$f_1 = K_p(q_{1f} - q_1) + K_i \int (q_{1f} - q_1)dt$$

$$f_2 = K_p e_2 + K_i \int e_2 dt$$

$$f_2 = K_p(q_{2f} - q_2) + K_i \int (q_{2f} - q_2)dt$$

3.4 Proportional Integral Derivative Controller

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

$$f = K_p e + K_d \frac{d(e)}{dt} + K_i \int e dt$$

By applying Laplace Transformation on above equation

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

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

$$f_1 = K_p e_1 + K_d \frac{d(e_1)}{dt} + K_i \int e_1 dt$$

$$f_1 = K_p(q_{1f} - q_1) + K_d \frac{d(q_{1f} - q_1)}{dt} + K_i \int (q_{1f} - q_1)dt$$

$$f_2 = K_p e_2 + K_d \frac{d(e_2)}{dt} + K_i \int e_2 dt$$

$$f_2 = K_p(q_{2f} - q_2) + K_d \frac{d(q_{2f} - q_2)}{dt} + K_i \int (q_{2f} - q_2)dt$$

4 Observation

The equations controlling the system dynamics of a two-link manipulator can be solved using a special inbuilt function known as `ode45`. The `ode45` function is MATLAB's standard solver for ordinary differential equations (ODEs). `ode45` is designed to handle the following general problem:

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

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

The general form of the system dynamics for a two-link manipulator can be written as:

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

where:

$M(\theta)$ = inertia matrix, $\quad C(\theta, \dot{\theta})$ = Coriolis/centrifugal matrix, $\quad G(\theta)$ = gravity vector

and τ is the vector of joint torques.

4.1 The Dynamic Code

```
function dqdt = Dynamics(t,q,m_1,m_2,l_1,l_2,g,Kp,Ki,Kd, ...
                         q_desired, e_int_prev, tspan)
   q1  = q(1);
   q2  = q(2);
   dq1 = q(3);
   dq2 = q(4);

  M11 = (m_1 + m_2)*l_1^2 + m_2*l_2*(l_2 + 2*l_1*cos(q2));
  M12 = m_2*l_2*(l_2 + l_1*cos(q2));
  M22 = m_2*l_2^2;

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

  C11 = -m_2*l_1*l_2*sin(q2)*dq2 ;
  C12 = -m_2*l_1*l_2*sin(q2)*(dq1+dq2);
  C21 = 0 ;

  C = [C11 , C12 ; C21 , -C11];

  G11 = m_1*l_1*g*cos(q1) + m_2*g*(l_2*cos(q1+q2) + l_1*cos(q1));
  G21 = m_2*g*l_2*cos(q1+q2);

  G = [G11;G21];

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


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

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

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



end
```

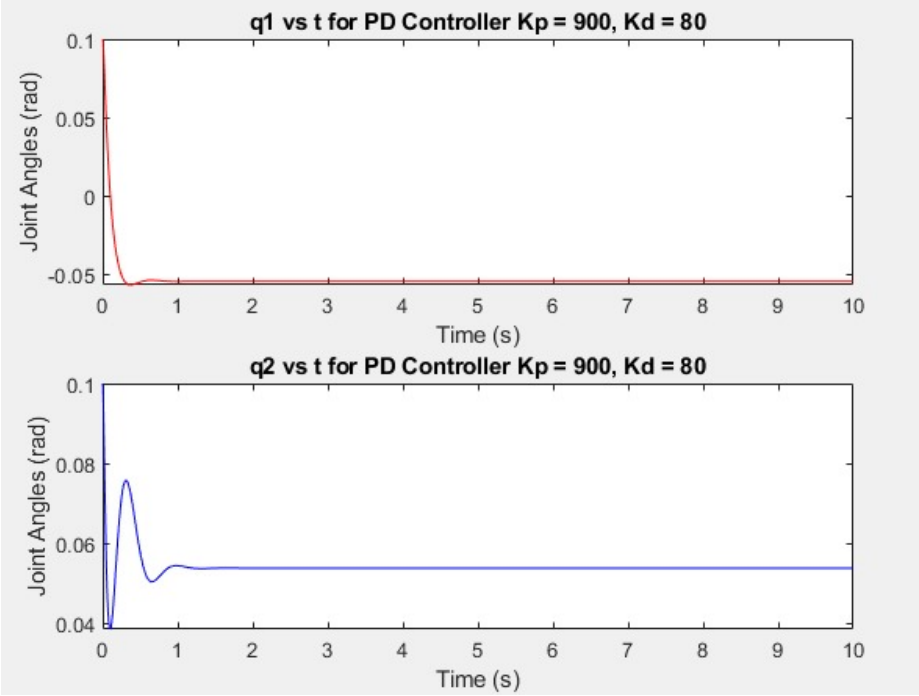
4.2 Proportional Derivative Controller

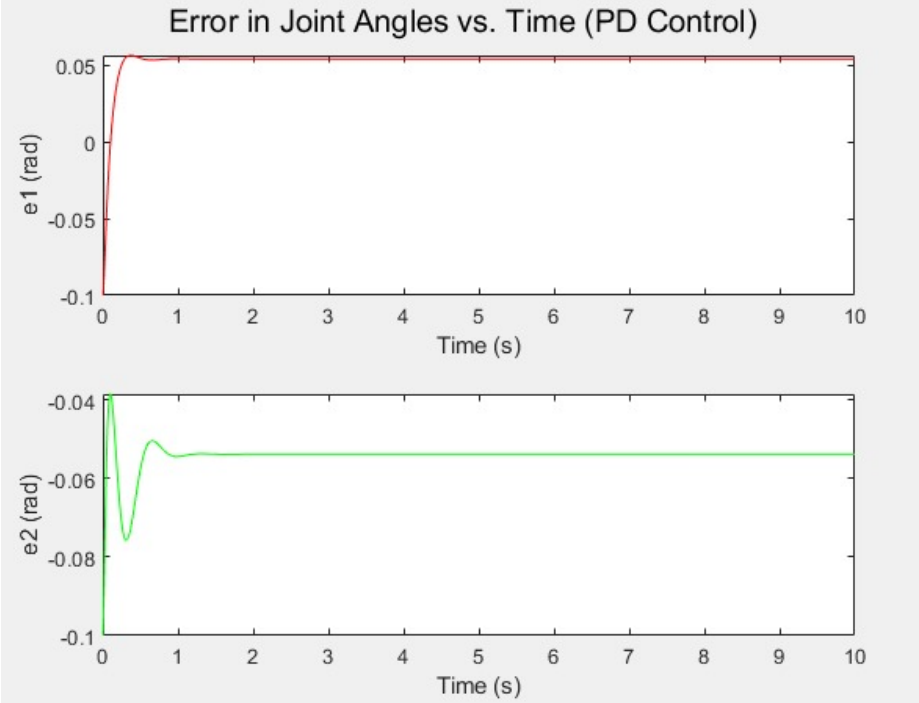
4.2.1 The Code

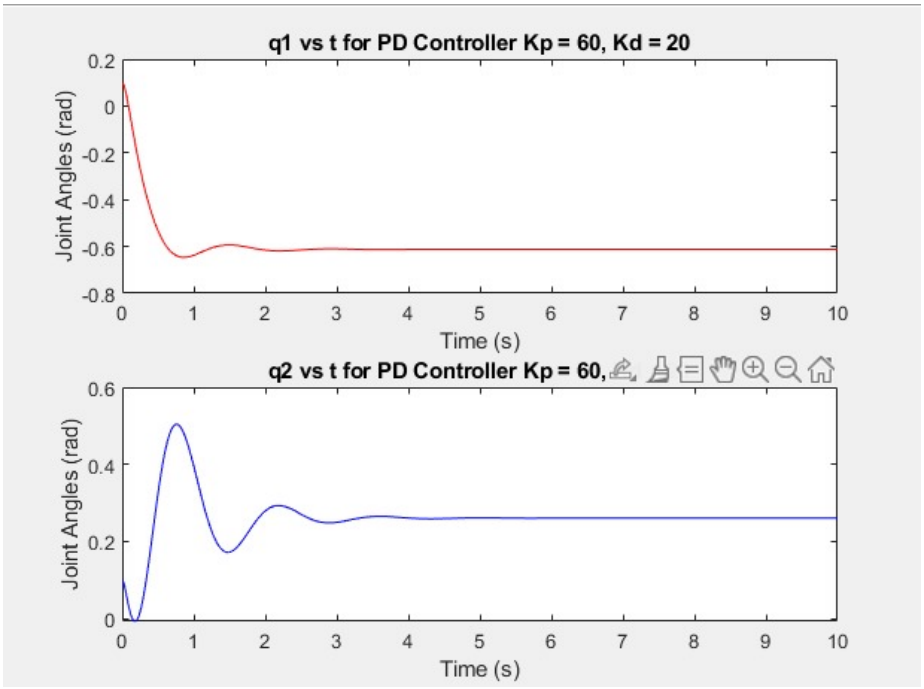
```
% for PD
m_1 = 10;
m_2 = 5;
l_1 = 0.2;
l_2 = 0.1;
```

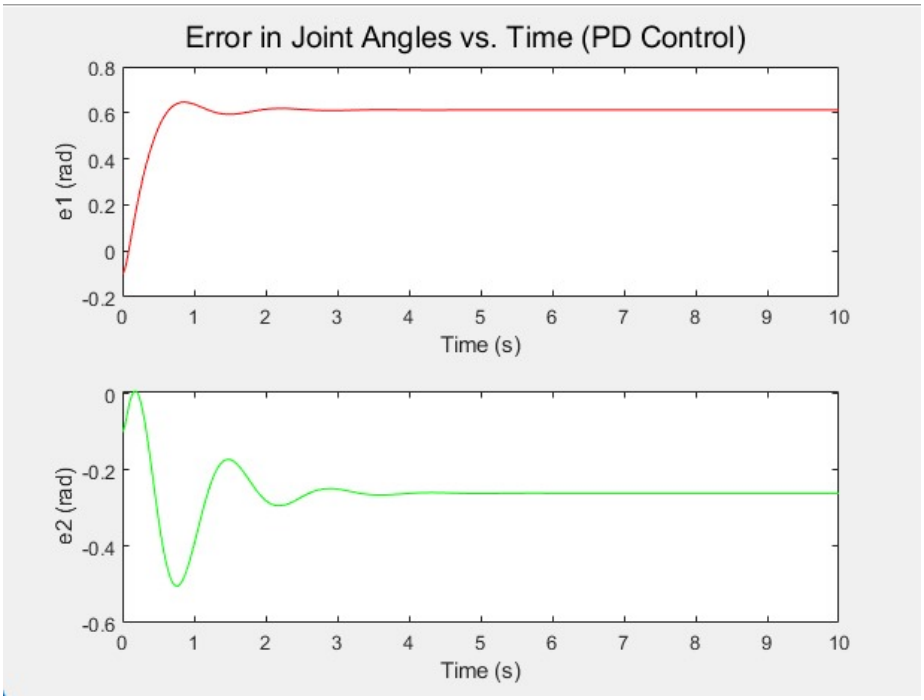
```
g = 9.81;

kp = 900;
kd = 80;
ki = 0;

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

q_initial = [0.1; 0.1; 0; 0];
q_desired = [0; 0];

tspan = [0 10];
e_int_prev = [0; 0];

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);
[t, q] = ode45(@(t, q) Dynamics(t, q, m_1, m_2, l_1, l_2, g, Kp, Ki, Kd, ...
                q_desired, e_int_prev, tspan), tspan, q_initial, options);

figure;
subplot(2,1,1);
plot(t, q(:, 1), 'r');
xlabel('Time (s)');
ylabel('Joint Angles (rad)');
title("q1 vs t for PD Controller Kp=" + kp + ", Kd=" + kd);

subplot(2,1,2);
plot(t, q(:, 2), 'b');
xlabel('Time (s)');
ylabel('Joint Angles (rad)');
title("q2 vs t for PD Controller Kp=" + kp + ", Kd=" + kd);

e1 = q_desired(1) - q(:,1);
e2 = q_desired(2) - q(:,2);
figure;
subplot(2, 1, 1);
plot(t, e1, 'r');
xlabel('Time (s)');
ylabel('e1 (rad)');
sgtitle('Error in Joint Angles vs. Time (PD Control)');
subplot(2, 1, 2);
plot(t, e2, 'g');
xlabel('Time (s)');
ylabel('e2 (rad)');
```

4.2.2 Plots Obtained






q1 vs t for PD Controller Kp = 60, Kd = 20
Joint Angles (rad)
Time (s)
q2 vs t for PD Controller Kp = 60,
Joint Angles (rad)
Time (s)


Error in Joint Angles vs. Time (PD Control)
e1 (rad)
Time (s)
e2 (rad)
Time (s)

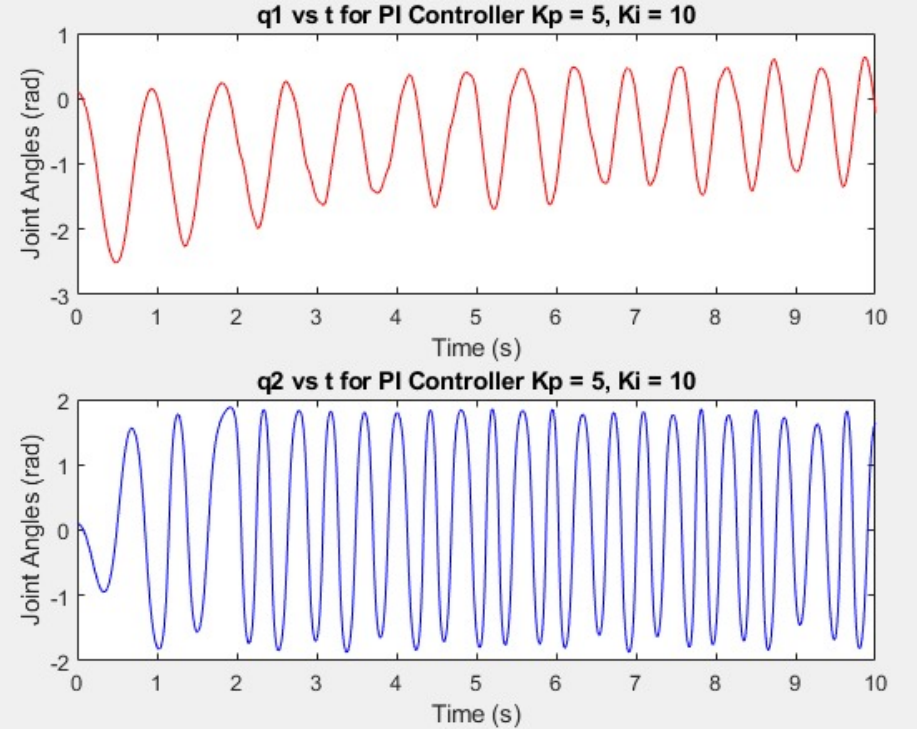
4.3 Proportional Integral Controller

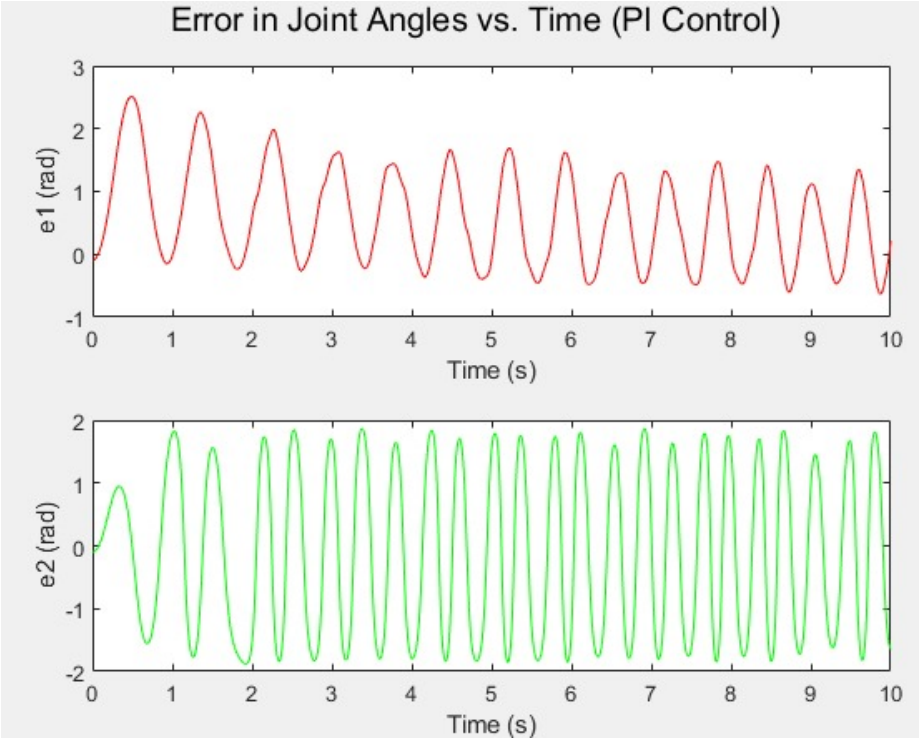
4.3.1 The Code

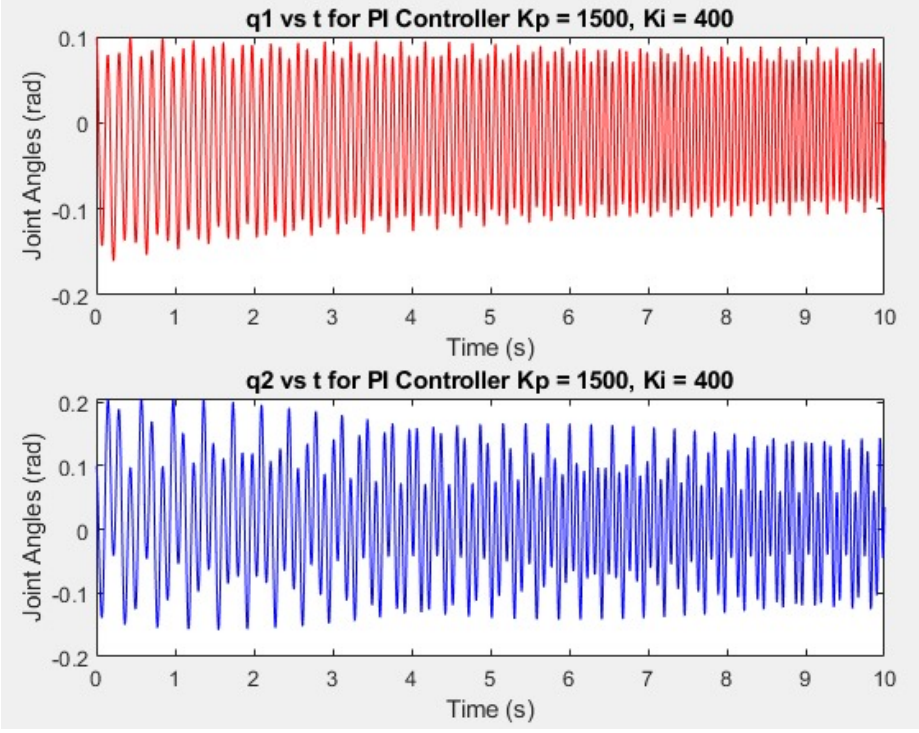
```
% for PI
m_1 = 10;
m_2 = 5;
l_1 = 0.2;
l_2 = 0.1;
g = 9.81;

kp = 250;
kd = 0 ;
ki = 3;

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

q_initial = [0.1; 0.1; 0; 0];
q_desired = [0; 0];

tspan = [0 10];
e_int_prev = [0; 0];

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);
[t, q] = ode45(@(t, q) Dynamics(t, q, m_1, m_2, l_1, l_2, g, Kp, Ki, Kd, ...
            q_desired, e_int_prev, tspan), tspan, q_initial, options);

figure;
subplot(2,1,1);
plot(t, q(:, 1), 'r');
xlabel('Time (s)');
ylabel('Joint Angles (rad)');
title("q1 vs t for PI Controller Kp=" + kp + ", Ki=" + ki);

subplot(2,1,2);
plot(t, q(:, 2), 'b');
xlabel('Time (s)');
ylabel('Joint Angles (rad)');
title("q2 vs t for PI Controller Kp=" + kp + ", Ki=" + ki);

e1 = q_desired(1) - q(:,1);
e2 = q_desired(2) - q(:,2);
figure;
subplot(2, 1, 1);
plot(t, e1, 'r');
xlabel('Time (s)');
ylabel('e1 (rad)');
sgtitle('Error in Joint Angles vs. Time (PI Control)');
```

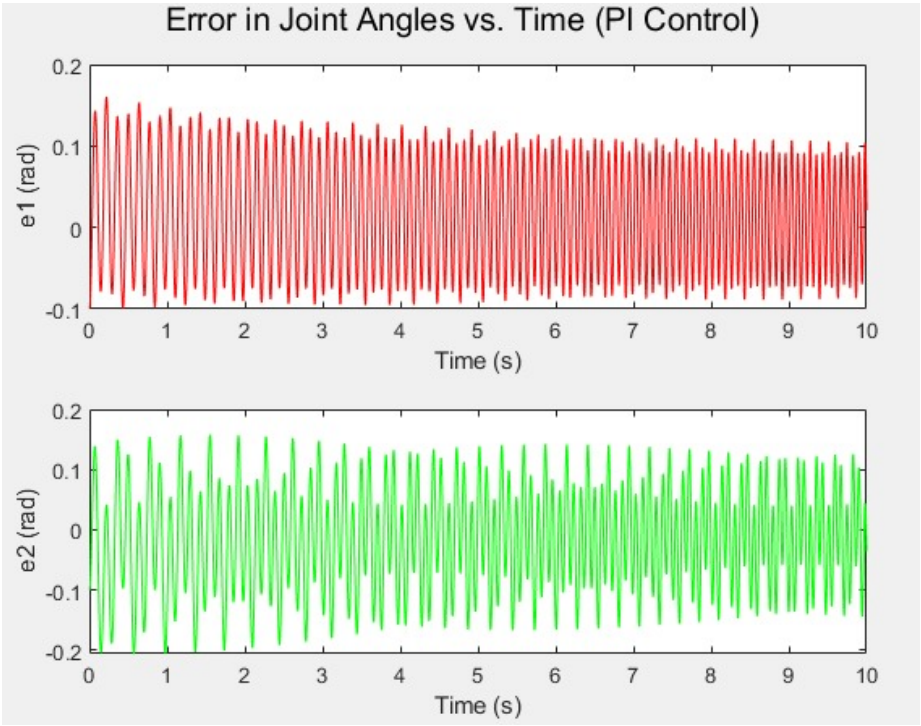
```
subplot(2, 1, 2);
plot(t, e2, 'g');
xlabel('Time (s)');
ylabel('e2 (rad)');
```

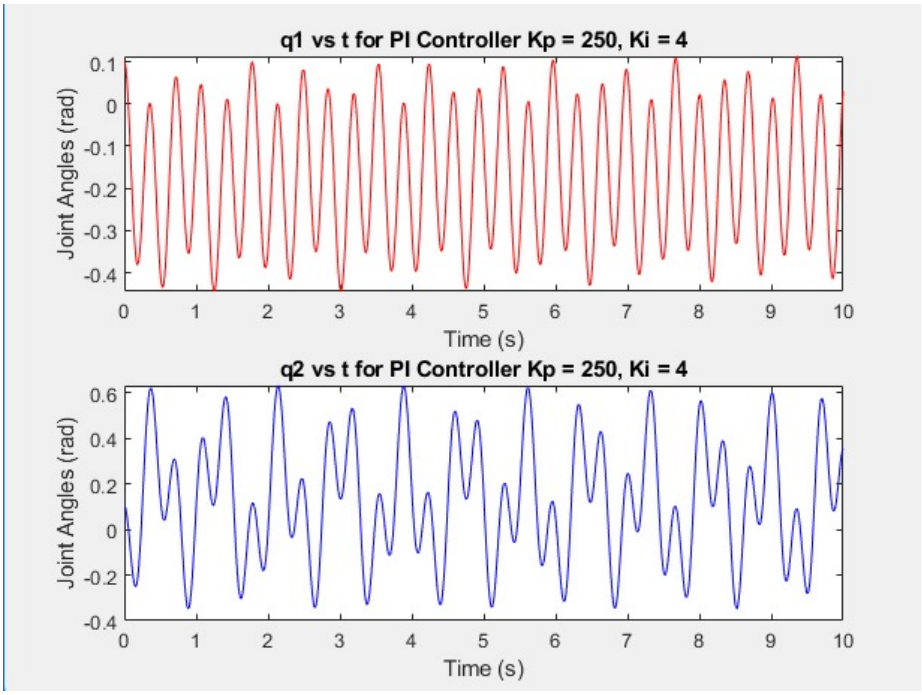
4.3.2 Plots Obtained

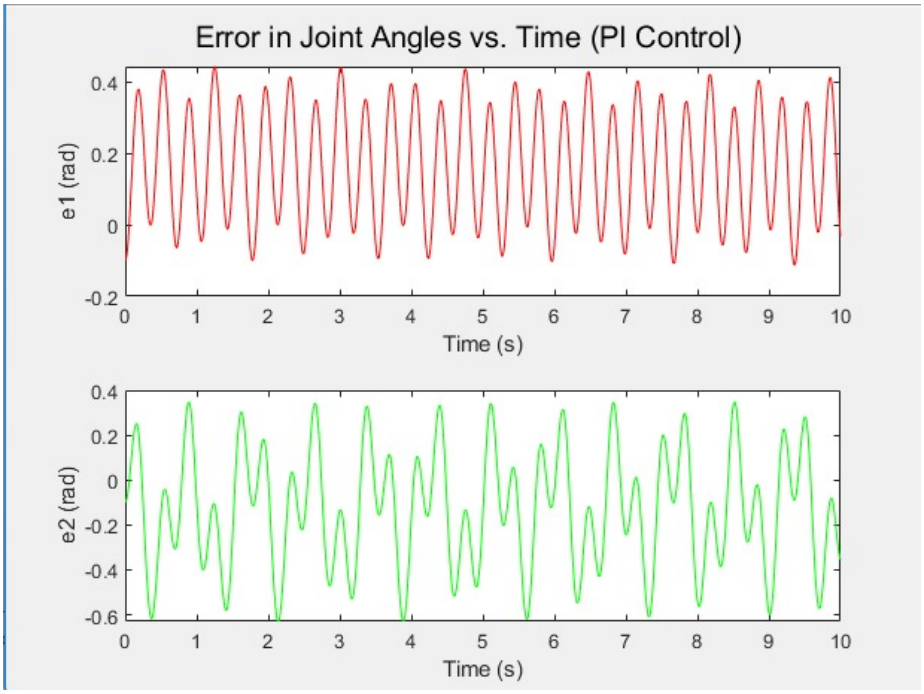



Error in Joint Angles vs. Time (PI Control)
e1 (rad)
Time (s)
e2 (rad)
Time (s)


q1 vs t for PI Controller Kp = 1500, Ki = 400
Joint Angles (rad)
Time (s)
q2 vs t for PI Controller Kp = 1500, Ki = 400
Joint Angles (rad)
Time (s)


Error in Joint Angles vs. Time (PI Control)
e1 (rad)
Time (s)
e2 (rad)
Time (s)


q1 vs t for PI Controller Kp = 250, Ki = 4
Joint Angles (rad)
Time (s)
q2 vs t for PI Controller Kp = 250, Ki = 4
Joint Angles (rad)
Time (s)



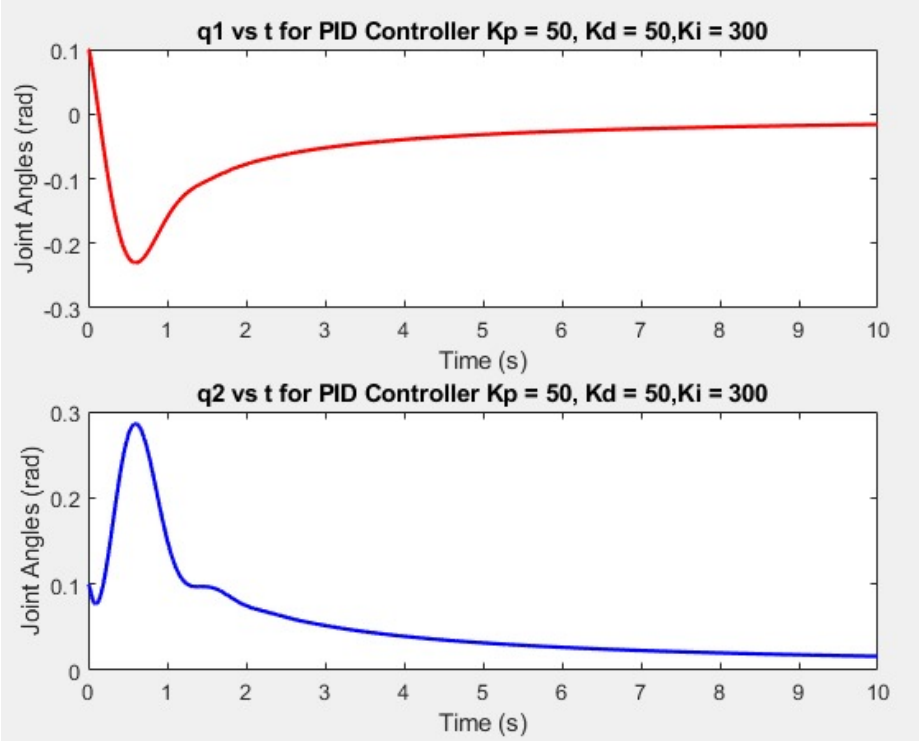
4.4 Proportional Integral Derivative Controller

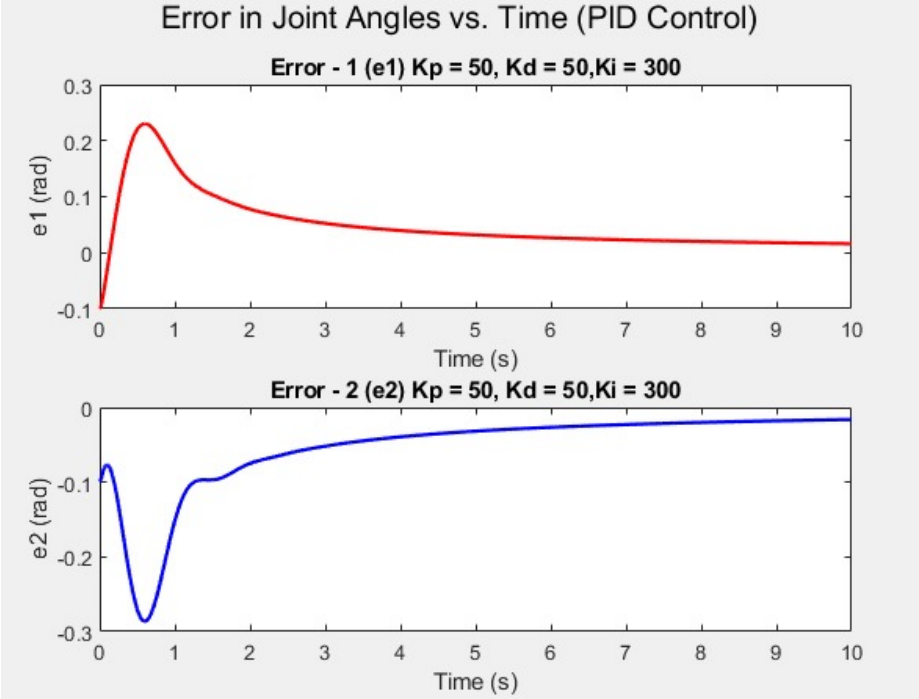
4.4.1 The Code

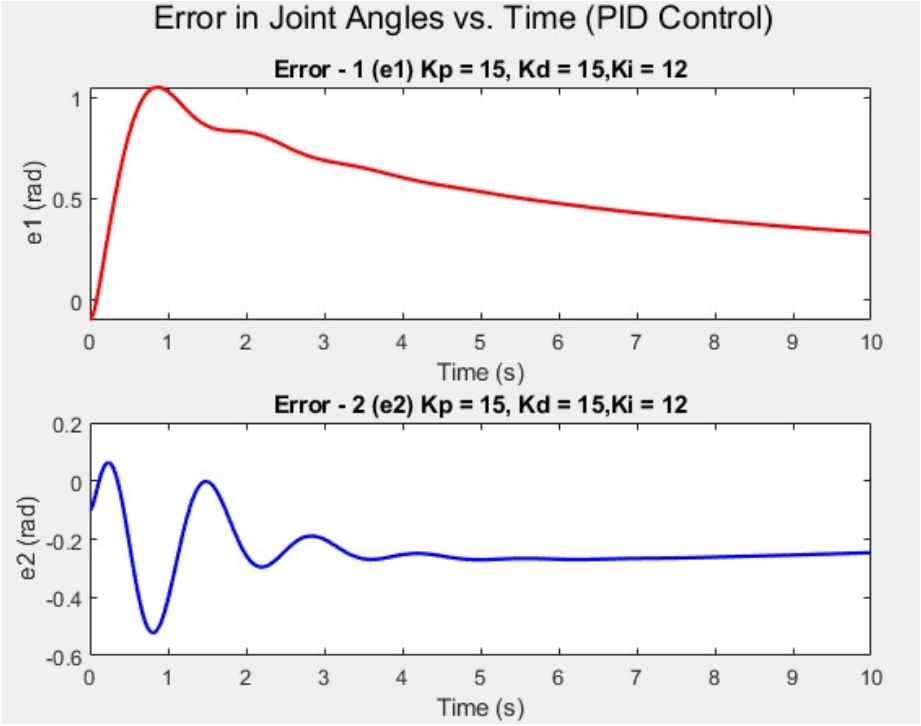
```
% for PD
m_1 = 10;
m_2 = 5;
l_1 = 0.2;
l_2 = 0.1;
g = 9.81;

kp = 50;
kd = 50;
ki = 300;

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

q_initial = [0.1; 0.1; 0; 0];
q_desired = [0; 0];

tspan = [0 10];
e_int_prev = [0; 0];

options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6);
[t, q] = ode45(@(t, q) Dynamics(t, q, m_1, m_2, l_1, l_2, g, Kp, Ki, Kd, ...
```

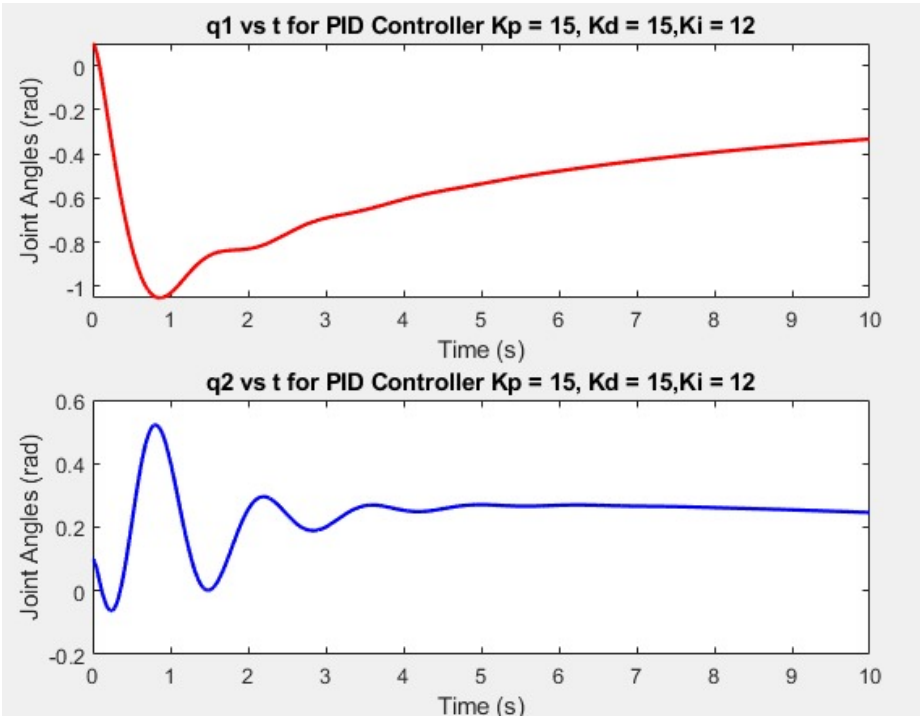
```
    q_desired, e_int_prev, tspan), tspan, q_initial, options);

figure;
subplot(2,1,1);
plot(t, q(:, 1), 'r');
xlabel('Time (s)');
ylabel('Joint Angles (rad)');
title("q1 vs t for PID Controller Kp=" + kp + ",Kd=" + kd + ",Ki=" + ki)

subplot(2,1,2);
plot(t, q(:, 2), 'b');
xlabel('Time (s)');
ylabel('Joint Angles (rad)');
title("q2 vs t for PID Controller Kp=" + kp + ",Kd=" + kd + ",Ki=" + ki)

e1 = q_desired(1) - q(:,1);
e2 = q_desired(2) - q(:,2);
figure;
subplot(2, 1, 1);
plot(t, e1, 'r');
xlabel('Time (s)');
ylabel('e1 (rad)');
sgtitle('Error in Joint Angles vs. Time (PID Control)');
subplot(2, 1, 2);
plot(t, e2, 'g');
xlabel('Time (s)');
ylabel('e2 (rad)');
```

4.4.2 Plots Obtained






Error in Joint Angles vs. Time (PID Control)
Error - 1 (e1) Kp = 15, Kd = 15,Ki = 12
e1 (rad)
Time (s)
Error - 2 (e2) Kp = 15, Kd = 15,Ki = 12
e2 (rad)
Time (s)

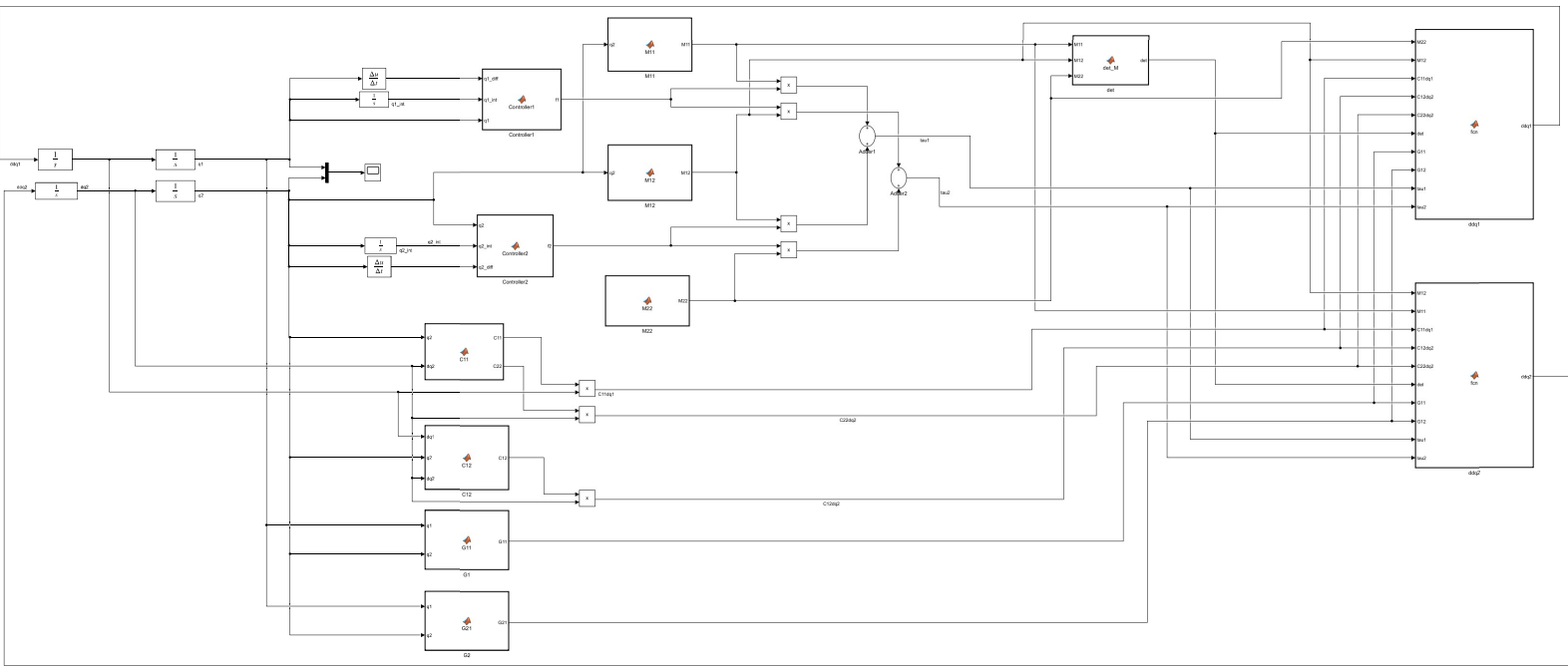

q1 vs t for PID Controller Kp = 15, Kd = 15,Ki = 12
Joint Angles (rad)
Time (s)
q2 vs t for PID Controller Kp = 15, Kd = 15,Ki = 12
Joint Angles (rad)
Time (s)

5 SIMULINK

The following describes the implementation of the system dynamics in Simulink:

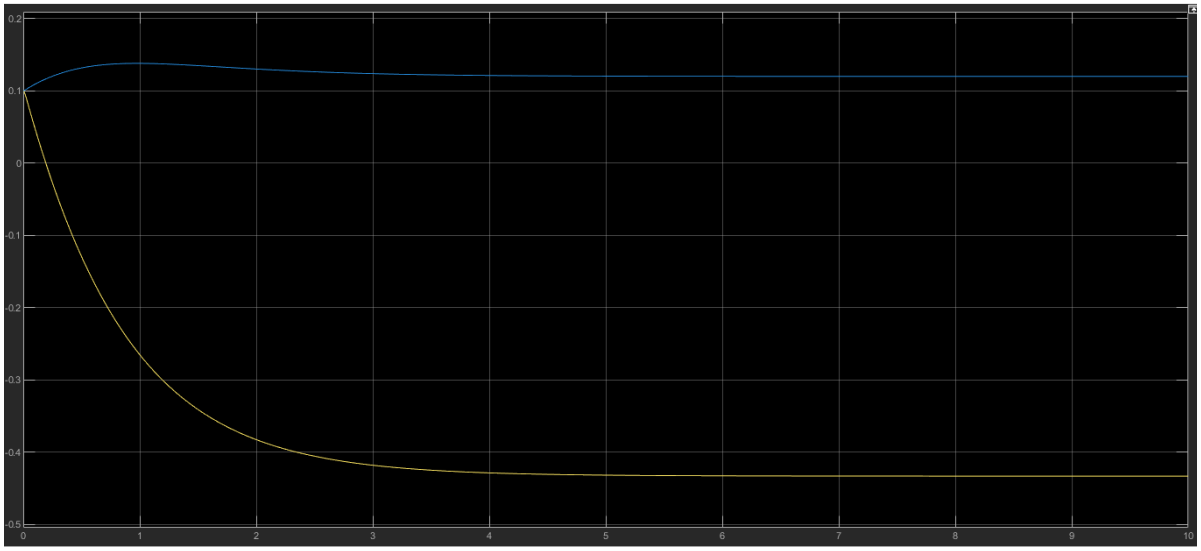
- The reference provides the desired final angles for the system.
- The controller computes the output value based on:
  - The constants of the system.
  - The error, which is the difference between the reference and the current angles.
- The calculated output, along with the states of the system, is provided to the state space subsystem block.
- The state space subsystem calculates the second-order derivative terms, which are a function of:
  - The current angles.
  - The first derivatives of the angles.
  - The output of the controller.
- The second-order derivatives are integrated using the integrator block:
  - The first integration yields the first-order derivatives (angular velocities).
  - The second integration gives the angles.
- The initial condition of the angles is provided inside the integrator block at the start of the simulation.
- The angle values are used to calculate the error by subtracting the current angles from the reference using a subtractor block.
- With the new values of the states (angles and their derivatives), the simulation proceeds by:
  - Using the state space subsystem block and integrator blocks to calculate the next state values.
  - The process continues over the time span specified for the simulation.
- The simulation starts with the initial conditions provided by the integrator block and runs until the specified time duration is reached.

5.1 Block Diagram

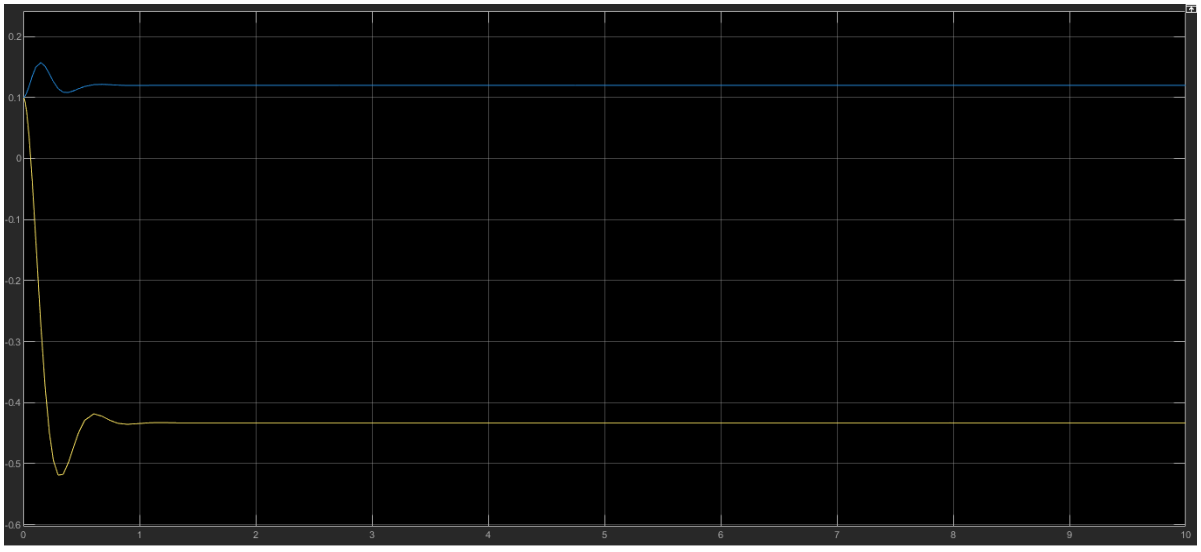

dot1
dot2
s1
s2
Controller1
Controller2
M11
M12
M22
det
Adder1
Adder2
tau1
tau2
C11
C12
C22
G1
G2
C11dq1
C22dq2
C12dq2
ddq1
ddq2

5.2 Proportional Derivative Controller

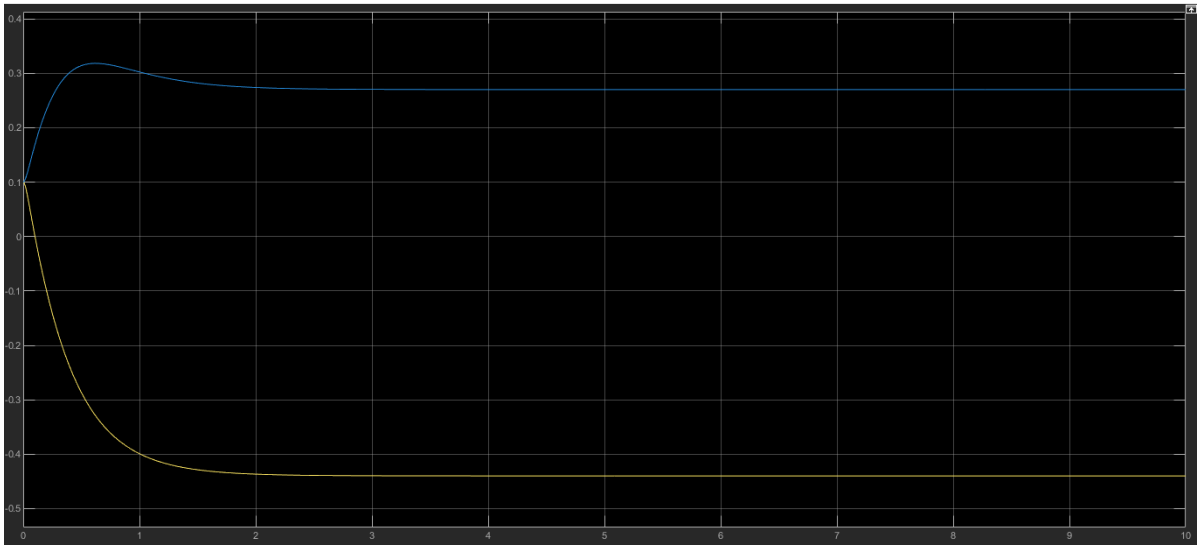
For $K_{p1} = 100$ $K_{D1} = 100$
For $K_{p2} = 300$ $K_{D2} = 200$



For $K_{p1} = 100$ $K_{D1} = 10$
For $K_{p2} = 300$ $K_{D2} = 20$

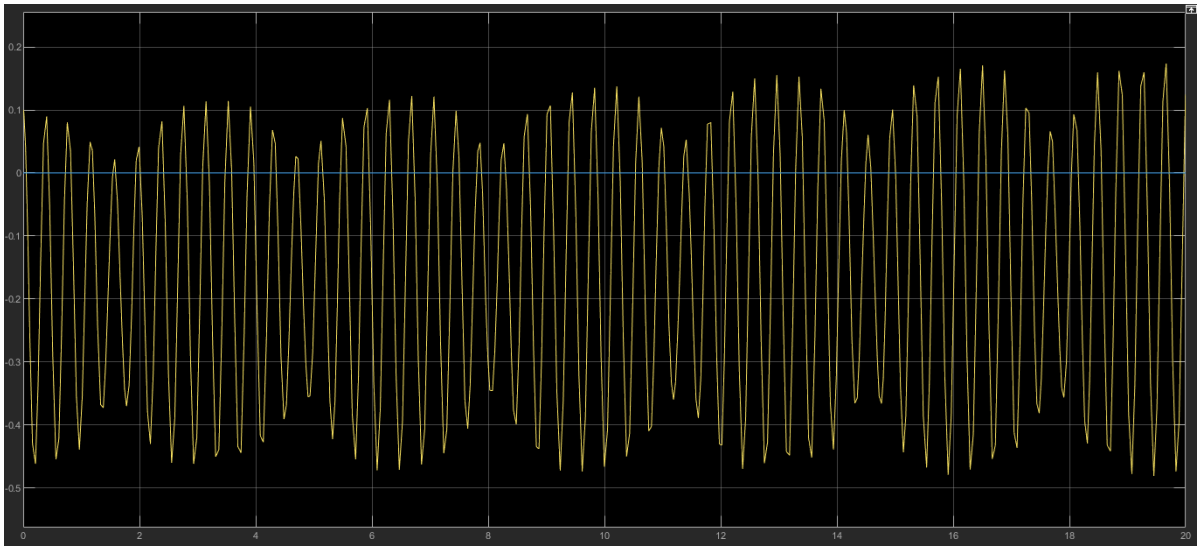


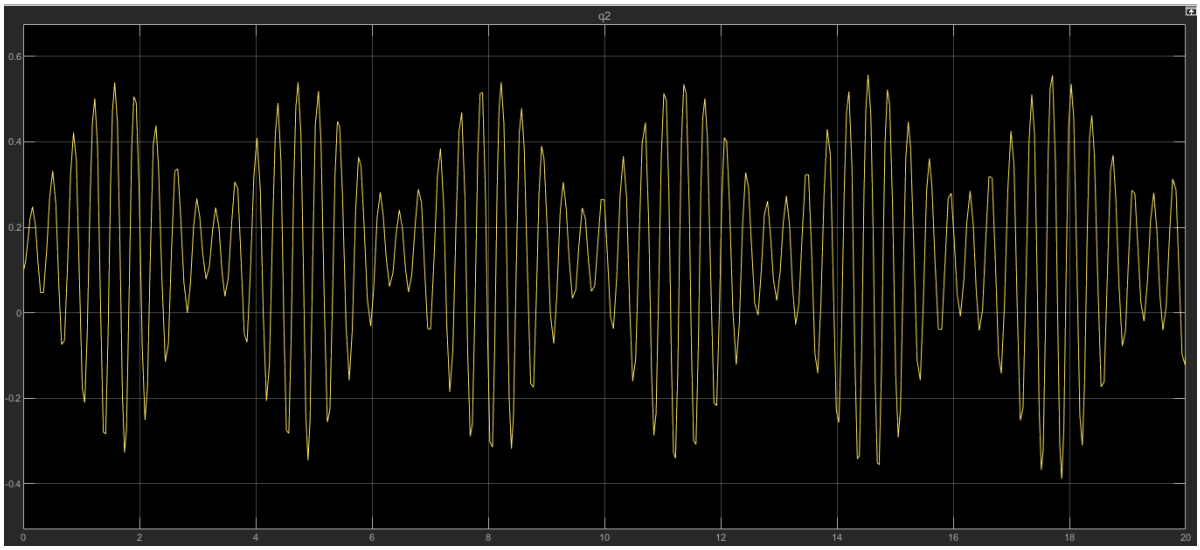
For $K_{p1} = 100$ $K_{D1} = 50$
For $K_{p2} = 100$ $K_{D2} = 50$



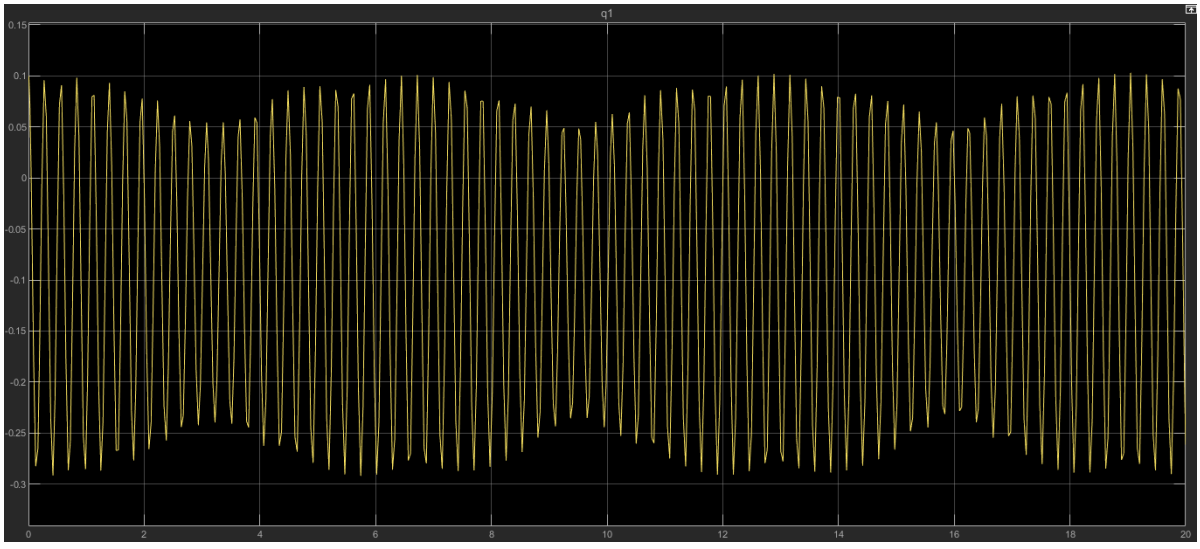
5.3 Proportional Integral Controller

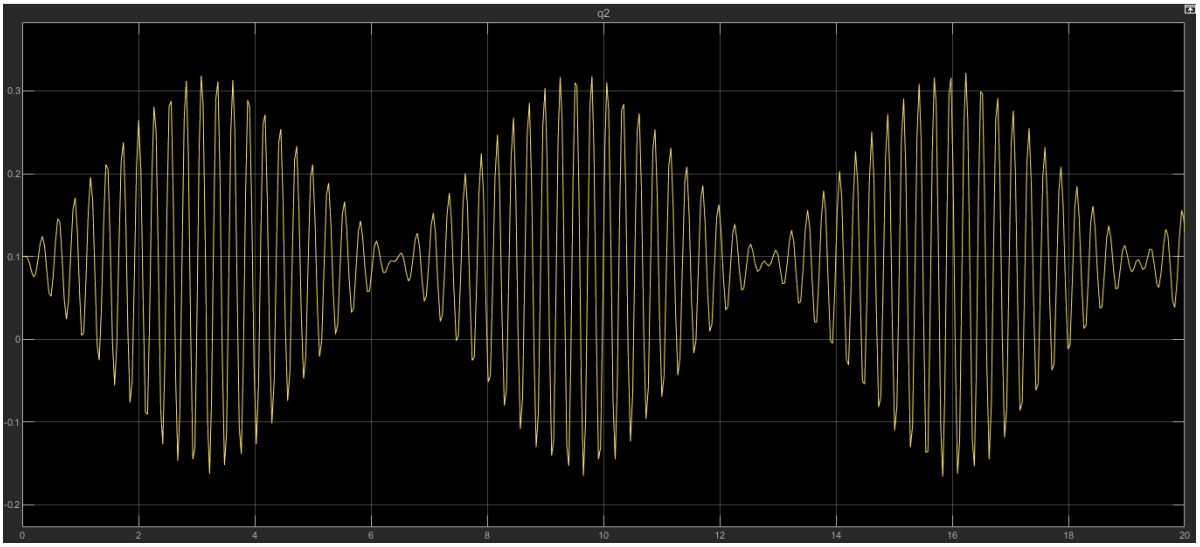
For $K_{p1} = 250$ $K_{I1} = 3$
For $K_{p2} = 250$ $K_{I2} = 3$





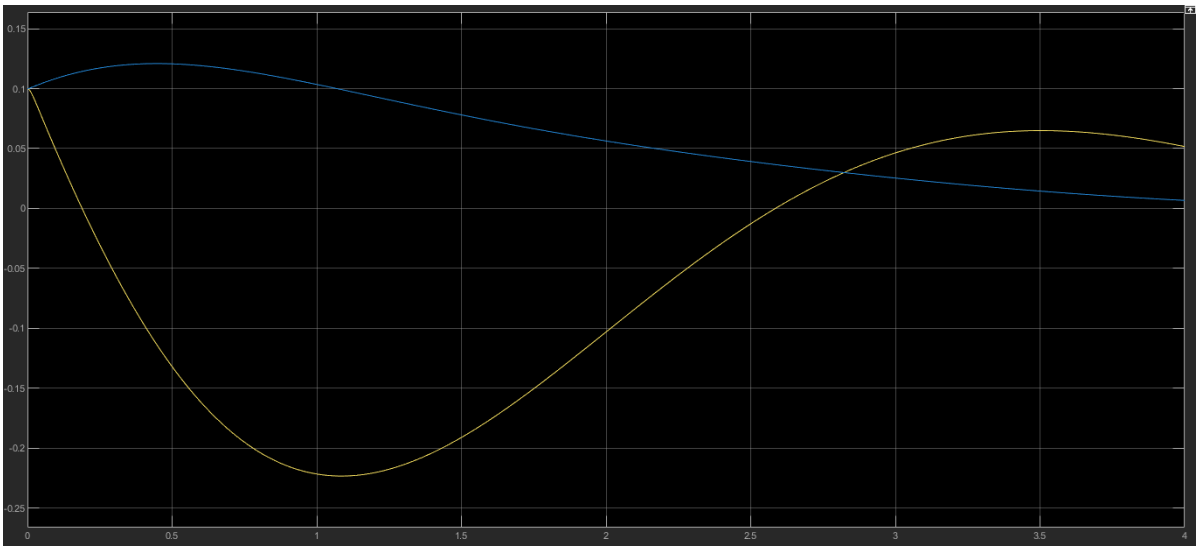
For $K_{p1} = 500$ $K_{I1} = 0.5$
For $K_{p2} = 500$ $K_{I2} = 0.5$



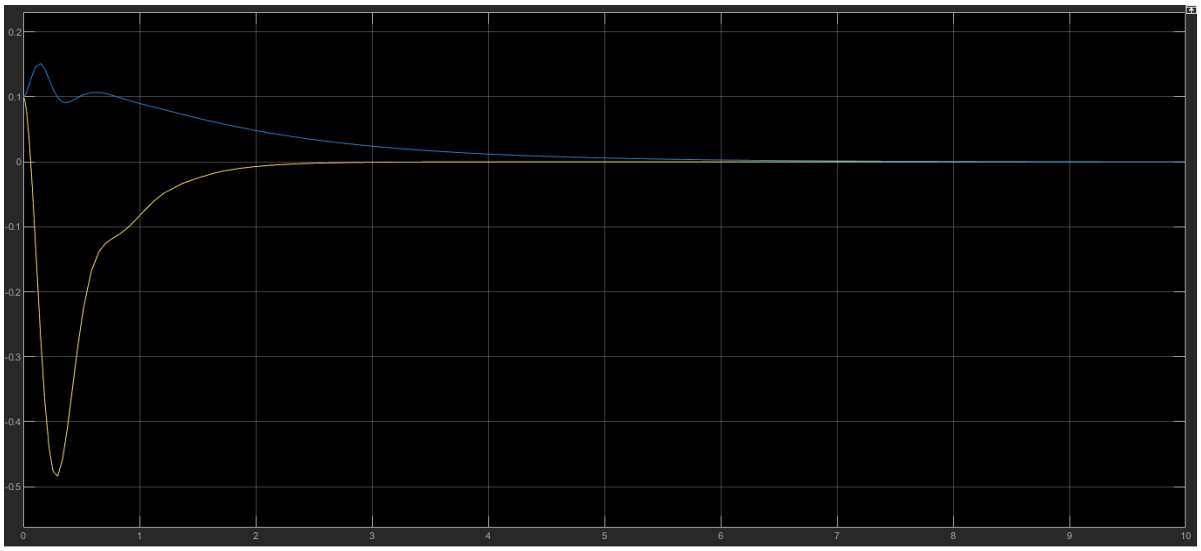


5.4 Proportional Integral Derivative Controller

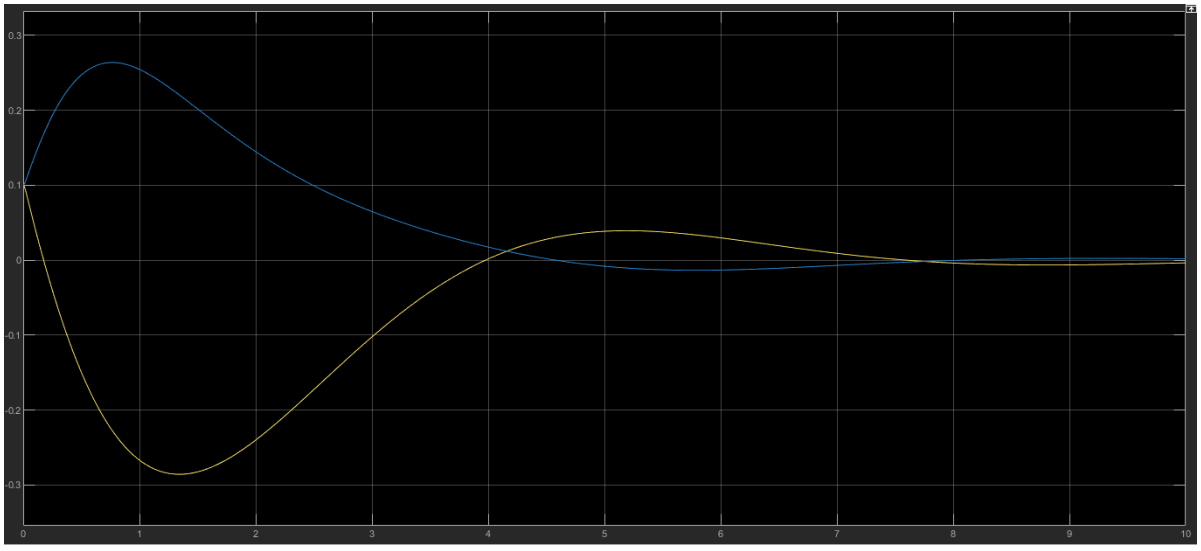
For $K_{p1} = 100$ $K_{D1} = 100$ $K_{I1} = 200$
For $K_{p2} = 300$ $K_{D2} = 200$ $K_{I2} = 200$



For $K_{p1} = 100$ $K_{D1} = 10$ $K_{I1} = 200$
For $K_{p2} = 200$ $K_{D2} = 20$ $K_{I2} = 200$



For $K_{p1} = 90$ $K_{D1} = 90$ $K_{I1} = 90$
For $K_{p2} = 90$ $K_{D2} = 90$ $K_{I2} = 90$



6 Conclusion

6.1 Proportional Derivative Controller

6.1.1 Influence on Damping

High K_D (Advantages):

- High derivative gain effectively reduces oscillations and minimizes the amplitude of oscillatory behavior, resulting in smoother and more stable control responses.
- It makes the robotic arm less susceptible to vibrations, which is critical when precise and controlled movements are required.

High K_D (Disadvantages):

- If the derivative gain is set too high, the system can become over-damped, leading to slower response times and reduced overall performance.
- Over-damping can also make the system less responsive, which may pose issues when rapid adjustments or dynamic movement are necessary.

Low K_D (Advantages):

- A lower K_D provides less damping, allowing for quicker movements in response to changes. This is advantageous in scenarios where fast reactions are prioritized over stability.
- The system becomes more agile, which is ideal for tasks requiring rapid and precise movements.

Low K_D (Disadvantages):

- Insufficient damping from a low K_D value may result in excessive oscillations, causing the system to overshoot and lose stability, which can negatively affect control accuracy.

6.1.2 Reducing Overshoot

High K_D (Advantages):

- A higher K_D helps in minimizing overshoot and oscillations, leading to a smoother and more controlled response. This is important when precision and stability are crucial.

High K_D (Disadvantages):

- Excessively high K_D values can cause over-damping, slowing down the system's settling time. This can hinder the system from reaching the target position quickly, which is problematic in time-sensitive operations.

Low K_D (Advantages):

- A low K_D can offer a faster initial response, which may allow some controlled overshoot. This can be useful when speed is more important than eliminating overshoot completely.

Low K_D (Disadvantages):

- Very low K_D values may not sufficiently reduce overshoot and could allow oscillations to persist, reducing the precision of the robotic arm's movements. This is a disadvantage when exact positioning is required.

6.1.3 Effect on Response Time

High K_D (Advantages):

- When set appropriately, a high K_D value can reduce the time it takes for the system to stabilize, improving the overall response time and making it more efficient for achieving accurate positioning.

High K_D (Disadvantages):

- In some cases, an excessively high KD may either cause overshoot or slow down the system's ability to settle, which affects performance when rapid movements are necessary.

Low K_D (Advantages):

- A lower K_D value maintains a fast initial response, though it might slightly increase the time needed for the system to fully stabilize. This is useful when initial speed is prioritized.

Low K_D (Disadvantages):

- Very low K_D can result in prolonged oscillations, delaying the settling process, which can be detrimental when both precision and stability are critical for the task.

6.1.4 Impact on System Stability

High K_D (Advantages):

- A high K_D contributes to a stable system when properly tuned, especially in scenarios that demand smooth, controlled, and precise movements without oscillations.

High K_D (Disadvantages):

- If the K_D value is too high, the system can become over-damped, leading to slow settling times and reducing the system's ability to respond effectively to dynamic changes. This could also result in potential instability under certain conditions.

Low K_D (Advantages):

- Lower K_D settings allow for a more responsive system by reducing the risk of over-damping, which makes it better suited for dynamic tasks where agility is required.

Low K_D (Disadvantages):

- Insufficient K_D may result in poor damping of oscillations, making the system less stable and prone to overshooting or erratic behavior, which could lead to instability.

6.1.5 Conclusion

Optimizing the derivative gain (K_D) in a PD control system involves striking a balance between minimizing overshoot, controlling oscillations, and maintaining the system's responsiveness. The ideal K_D setting depends on the mechanical properties of the system, the required performance, and the specific tasks being performed. Careful tuning is essential to ensure that the robotic arm operates efficiently, providing the desired balance between speed, precision, and stability.

6.2 Proportional Integral Controller

6.2.1 Impact of Proportional Gain (K_p)

High K_p (Advantages):

- Increasing K_p enhances the controller's responsiveness to deviations in joint angles, allowing for more aggressive correction when the system deviates from its desired position.
- A higher K_p can lead to quicker convergence to the target joint angles, even in the presence of significant errors.

High K_p (Disadvantages):

- An excessively high K_p can cause the system to overshoot the desired position, potentially resulting in oscillations or instability, which can complicate precise control.
- High K_p may increase sensitivity to noise and disturbances, leading to erratic or unpredictable behavior.

Low K_p (Advantages):

- A lower K_p provides a more gradual and stable response to errors, which can be useful when stability is prioritized over speed.
- It reduces the likelihood of overshoot and oscillations, creating a smoother and more controlled system response.

Low K_p (Disadvantages):

- A very low K_p might result in sluggish performance, with slow convergence to the target position, especially when there are larger errors.
- It may delay achieving the desired joint angles, which could be problematic in time-sensitive applications.

6.2.2 Impact of Integral Gain (K_i)

High K_i (Advantages):

- Increasing K_i improves the controller's ability to eliminate steady-state errors by integrating the error over time, ensuring that the system reaches and maintains the desired joint angles.

- A high K_i can increase the system's accuracy, making it settle closer to the target position with fewer long-term deviations.

High K_i (Disadvantages):

- Excessive K_i can introduce instability, especially in noisy environments or where disturbances are present. This could result in overshooting, oscillations, or erratic system behavior.
- High K_i values often require careful tuning and may not be suitable for all systems or conditions.

Low K_i (Advantages):

- A lower K_i provides a more stable response with reduced sensitivity to noise or external disturbances, decreasing the likelihood of oscillations or overshoot.
- Lower values of K_i are suitable when steady-state error correction isn't critical.

Low K_i (Disadvantages):

- Very low K_i might allow steady-state errors to persist, reducing the system's ability to maintain accurate joint angles over time, especially in applications that require precision.

6.2.3 Steady-State Error Correction

High K_i (Advantages):

- High K_i effectively reduces steady-state errors, ensuring that the system converges closely to the target joint angles without long-term deviations.

High K_i (Disadvantages):

- Too much K_i can destabilize the system, increasing the risk of overshoot and oscillations, particularly in systems with significant noise or disturbances.

Low K_i (Advantages):

- Lower K_i values create a more stable system but might allow small steady-state errors to remain.

Low K_i (Disadvantages):

- Very low K_i may not sufficiently correct steady-state errors, potentially leading to reduced system accuracy over time.

6.2.4 Conclusion

Tuning a PI controller requires balancing K_p and K_i to achieve the desired performance while avoiding instability, overshooting, or oscillations. The optimal gains depend on the specific characteristics of the robotic arm system and the control requirements of the application.

6.3 Proportional Integral Derivative Controller

6.3.1 Effect of Proportional Gain (K_p)

High K_p (Advantages):

- **Reduced steady-state error**: A high proportional gain quickly reduces the steady-state error by scaling the control action proportionally to the current error. This results in a stronger control response when the error is significant, allowing the system to approach the desired position more rapidly.
- **Enhanced accuracy**: Higher values of K_p improve precision and positioning accuracy, making it suitable for applications requiring fine control, such as robotic arms.

High K_p (Disadvantages):

- **Risk of oscillations**: Excessively high K_p values can induce oscillatory behavior or instability. This makes it challenging to achieve smooth and precise control.
- **Overshoot and erratic behavior**: When K_p is too high, the system may overshoot the desired position before settling, potentially leading to an unstable or erratic response.

Low K_p (Advantages):

- **Improved stability**: Lower K_p values reduce the likelihood of oscillations and instability, leading to a more controlled and steady response to errors.

Low K_p (Advantages):

- **Slower correction**: Low K_p results in a less aggressive control response, which slows down error correction. As a result, the system may take longer to reach the desired position.

6.3.2 Effect of Derivative Gain (K_d)

High K_d(Advantages):

- **Dampening oscillations**: A high derivative gain effectively reduces the rate of change of the error, dampening oscillations and ensuring a smoother system response.
- **Increased stability**: Proper tuning of K_d leads to greater stability, making it ideal for applications requiring high precision and steady control, like in robotic systems.

High K_d(Disadvantages):

- **Over-damping**: High K_d can lead to an over-damped response, where the system becomes sluggish and slow to settle. This may hinder the system's ability to adapt to dynamic changes, affecting its responsiveness.

Low K_d(Advantages):

- **Faster initial response**: A low K_d value allows the system to quickly react to errors and rapidly approach the desired position.

Low K_d(Disadvantages):

- **Reduced damping**: Very low K_d values may be insufficient to dampen oscillations effectively, potentially compromising system stability.

6.3.3 Effect of Integral Gain (K_i)

High K_i (Advantages):

- **Addresses persistent errors**: A high integral gain is effective at eliminating long-term, persistent errors by accumulating past errors and adjusting the control output accordingly. This ensures that the system reaches the target state despite any steady-state disturbances.

High K_i (Disadvantages):

- **Instability risk**: If K_i is too high, it can lead to instability, especially in systems prone to noise or disturbances. This may result in unpredictable and erratic control behavior.

Low K_i (Advantages):

- **Stability preservation**: Low K_i values avoid excessive accumulation of errors, reducing the risk of oscillations and maintaining overall system stability. This leads to smoother and more gradual control.

Low K_i (Disadvantages):

- **Steady-state error**: Insufficient K_i can fail to eliminate steady-state errors, leaving persistent inaccuracies in the system's performance.

6.3.4 Conclusion

Tuning a PID controller involves striking a balance between K_p, K_i, and K_d to achieve optimal control performance. The goal is to minimize steady-state error, avoid overshooting or oscillations, and maintain system stability. The ideal tuning parameters will vary based on the specific dynamics of the robotic arm and the requirements of the application.