Two Link Robotic Manipulator

The Better Half

Member One
RollNo

Member Two
RollNo

Member Three
RollNo

Member Four
RollNo

Member Two
RollNo

Member Two
RollNo

Member Four
RollNo

Member Four
RollNo

October 7, 2023

Abstract

This report explores the use of state functions and variables to model a 2-link robotic manipulator's dynamics. The goal is to transition the system from its initial to final states while optimizing control. In the study, we implement PI, PD, and PID controllers to manage the manipulator's complex, non-linear behaviour. The emphasis is on precise position control through torque modulation. We formulate the equations of motion and carry out simulations in MATLAB-Simulink to validate our approach

Contents

1 Introduction

Robotic manipulators play a critical role in a diverse array of applications, from industrial assembly lines to surgical operations. The 2-link manipulator serves as a fundamental model in the field of robotics, offering both simplicity and a basis for understanding more complex systems.

1.1 Conceptual Modeling

We can approximate the robotic arm as a combination of two mass-less rods, each having a mass at its end. In this model, one rod-mass pair acts as the robot's arm, while the other functions as the forearm. Additional masses are assumed to be placed at the elbow joint and at the forearm's end. This arrangement allows us to couple these two systems and analyze the overall arm dynamics. The systems are free to oscillate within a plane.

1.2 Theoretical Framework

The manipulator's dynamics can be comprehensively described by formulating its Lagrangian, which is a function of its potential and $K_i$netic energies. Using the Euler-Lagrange equation, we can derive the torques applied to each link. Subsequently, we employ PI, PD, or PID controllers to simulate the system's behaviour under varying torque applications.

1.3 Methodology

The equation of motion for this 2-link manipulator is a non-linear differential equation. We solve it numerically and employ MATLAB for simulation and visualization of the control mechanisms.

1.4 Objectives

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

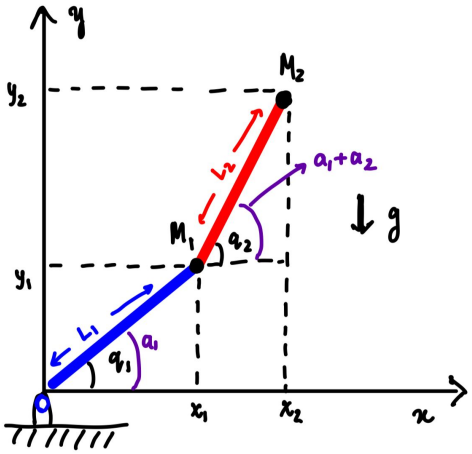
The remainder of this paper is organized as follows: Section 2 details the mathematical modelling, Section 3 discusses the controller designs, Section 4 presents the simulation results, and Section 5 concludes the study with future recommendations.

2 Dynamics of a 2-link Manipulator

A two-link manipulator can be simulated by considering a system with two mass-less rods of length L_1 and L_2 with masses M_1 and M_2 attached to the end of each rod, respectively. Here, M_1 acts as the elbow joint, while M_2 serves as the end of the forearm when modelling the robotic arm as a 2-link manipulator.

Let the robotic arm be attached to a surface. then let q_1 be the angle L_1 makes with the surface, and q_2 be the angle L_2 makes with the surface. We can represent this system on a 2D plane.

In this plane, we can define the x and y positions of each of the masses M_1 and M_2 respectively.



A simplified model of a two-link planar robot manipulator

After we use Lagrangian formulation on this system, we find the system equations by solving for four state variables. We can represent our system

by this equation:

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

Where:

- $M(a)$ is a positive-definite $n \times n$ matrix representing manipulator inertia. It depends on the joint angles a_1 and a_2.
- $C(a, \dot{a})$ is a $n \times n$ matrix representing the Coriolis and centrifugal torques. It depends on the joint angles a_1 and a_2 and the joint velocities $\dot{a}_1$ and $\dot{a}_2$.
- τ is an $n \times 1$ matrix representing the torques applied at each joint.
- $\ddot{a}$ and $\dot{a}$ are, respectively, the $n \times 1$ matrices of joint accelerations and joint velocities.

In this paper, since we are formulating for a 2-link manipulator, we can take n = 2.

After Lagrangian formulation, applying the Euler-Lagrangian equation, and further manipulation, we get the following values for the different matrices in our equation of motion. (We have made the following substitutions to simplify the math: $q_1 = a_1$ and $q_2 = a_1 + a_2$. In the following sections, we go back to the usage of q for simplicity.) The dynamics are described by:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{12} & M_{22} \end{bmatrix}, \quad a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \tag{1}$$

Where:

$$M_{11} = (m_1 + m_2)l_1^2 + m_2 l_2 (l_2 + 2l_1 \cos(a_2)), \tag{2}$$

$$M_{12} = m_2 l_2 (l_2 + l_1 \cos(a_2)), \tag{3}$$

$$M_{22} = m_2 l_2^2 \tag{4}$$

$$C = \begin{bmatrix} -m_2 l_1 l_2 \sin(a_2)\dot{a}_2 & -m_2 l_1 l_2 \sin(a_2)(\dot{a}_1 + \dot{a}_2) \\ 0 & m_2 l_1 l_2 \sin(a_2)\dot{a}_2 \end{bmatrix}, \tag{5}$$

$$G = \begin{bmatrix} m_1 l_1 g \cos(a_1) + m_2 g (l_2 \cos(a_1 + a_2) + l_1 \cos(a_1)) \\ m_2 g l_2 \cos(a_1 + a_2) \end{bmatrix} \tag{6}$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{7}$$

3 Control Design of a 2-link Manipulator

In the control of 2-link robotic manipulators, minimizing errors in position and velocity is crucial for achieving precise and stable motion. One of the widely used approaches to accomplish this is by employing Proportional-Integral (PI) or Proportional-Derivative (PD) or Proportional-Integral-Derivative (PID) controllers.

- **Proportional Gain (K_p)**: The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant known as K_p, the proportional gain constant.
- **Integral Gain (K_i)**: The integral term is concerned with the accumulation of past errors. If a small error has been present for an extended period, summing it up produces a large drive signal. This drive signal moves the system towards a smaller error. This term has an effect on the final error.
- **Derivative Gain (K_d)**: The derivative term is a prediction of future error, based on its rate of change. It provides a control output to counteract the rate of error change. The contribution of the derivative term to the overall control action is termed as K_d. This term helps reduce overshoot and ringing and has no effect on final error.

These gains (K_p, K_i, and K_d) are tuned to get the optimal system performance, and they play a significant role in reducing the steady-state and transient errors, leading to more accurate and stable control of the 2-link manipulator.

We can define our error variable P_{error} as P_{set} - P_{sensor} where P_{set} gives us the position we want to go to, and P_{sensor} gives us the position that we actually are at. $P_{error} = P_{set}$ - P_{sensor}

From the previous section, we know that the equation of motion can be written as follows:

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

where the matrices are defined according to equations (1), (2), (3), (4), (5), (6), (7). Solving for $\ddot{q}$:

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

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

We can take $\hat{F}$ as the new input where:

$$\hat{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{8}$$

knowing that our original physical input can be written as:

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{9}$$

Here we take $\hat{F}$ as our control signal. Our goal with any control design is to minimize our error signal using our control signal. Let us denote our error signals like this

$$e(q_1) = q_{1f} - q_1 \tag{10}$$

and

$$e(q_2) = q_{2f} - q_2 \tag{11}$$

Here we assume the target positions of M_1 and M_2 are given by the angles q_{1f} and q_{2f} respectively.

Let the below matrix denote the initial positions of our system:

$$q_o = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{12}$$

3.1 Using a PI Controller

In a PI (Proportional-Integral) controller, the controller uses two responses, namely the proportional and integral terms, to generate the control signal that minimizes the error. Here's how you can express the equation for a PD controller

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

The technique which we can employ for controlling the double joint system with inputs f_1 and f_2 is to employ two independent controllers, one for each link. We can write the following equations:

$$f_1 = K_{P_1} e_1(q_1) + K_{I_1} \int e(q_1)\, dt$$
$$= K_{P_1}(q_{1f} - q_1) + K_{I_1} \int (q_{1f} - q_1)\, dt$$

$$f_2 = K_{P_2} e_2(q_2) + K_{I_2} \int e(q_2)\, dt$$
$$= K_{P_2}(q_{2f} - q_2) + K_{I_2} \int (q_{2f} - q_2)\, dt$$

The complete system of equations with control is then

$$\ddot{q} = -M^{-1}(q)[c(q, \dot{q}) + G(q)] + \hat{F}$$

$$\hat{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} K_{P_1}(q_{1f} - q_1) + K_{I_1} \int (q_{1f} - q_1)\, dt \\ K_{P_2}(q_{2f} - q_2) + K_{I_2} \int (q_{2f} - q_2)\, dt \end{bmatrix}$$

.

The actual physical torques are,

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

.

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = -M^{-1}(q)[c(q, \ddot{q}) + G(q)] + \begin{bmatrix} K_{P_1}(q_{1f} - q_1) + K_{I_1} \int (q_{1f} - q_1)\, dt \\ K_{P_2}(q_{2f} - q_2) + K_{I_2} \int (q_{2f} - q_2)\, dt \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} K_{P_1}(q_{1f} - q_1) + K_{I_1} \int (q_{1f} - q_1)\, dt \\ K_{P_2}(q_{2f} - q_2) + K_{I_2} \int (q_{2f} - q_2)\, dt \end{bmatrix} \tag{14}$$

We now discretise the differential equations above by transforming them into a first-order ordinary differential equation system. This is accomplished by introducing new variables:

$$u_1 = q_1,\ u_2 = q_2,\ u_3 = \int (q_{1f} - q_1)\, dt\, u_4 = \int (q_{2f} - q_2)\, dt, \tag{15}$$

$$\dot{u_1} = \dot{q_1}$$

$$\dot{u_2} = \dot{q_2}$$

$$\dot{u_3} = q_{1f} - q_1$$

$$\dot{u_4} = q_{2f} - q_2$$

where ϕ and ψ are expressed in terms of u:

$$\begin{bmatrix}\phi\\ \psi\end{bmatrix} = -M^{-1}(q)[c(q,\ddot{q}) + G(q)] + \begin{bmatrix}K_{P_1}(q_{1f} - u_1) + K_{I_1}u_3\\ K_{P_2}(q_{2f} - u_2) + K_{I_2}u_4\end{bmatrix}$$

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

Now, we have a system of first-order non-linear differential equations of the firm:

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

where,

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

$$H = \begin{bmatrix}q_{1f} - u_1\\ q_{2f} - u_2\\ \phi(t, u_1, u_2, u_3, u_3, u_4)\\ \psi(t, u_1, u_2, u_3, u_3, u_4)\end{bmatrix}$$

. The initial conditions are

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

This is solved using the ode45 command in Matlab(see code). This gives us the torque.

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = M(q)\begin{bmatrix}K_{P_1}(q_{1f} - u_1) + K_{I_1}u_3\\ K_{P_2}(q_{2f} - u_2) + K_{I_2}u_4\end{bmatrix} \tag{16}$$

Equating and rearranging, we get,

$$\tau_1 = (M_1 + M_2)L_1^2(K_{P_1}(q_{1f} - u_1) + K_{I_1}u_3 +$$

$$+M_2L_1L_2\cos(u_1 - u_2)(K_{P_2}(q_{2f} - u_2) + K_{I_2}u_4,$$

$$\tau_2 = M_2L_1L_2\cos(u_1 - u_2))(K_{P_1}(q_{1f} - u_1) + K_{I_1}u_3$$
$$+M_2L_2^2(K_{P_2}(q_{2f} - u_2) + K_{I_2}u_4$$

.

3.1.1 Advantages

PI controller is used to reduce both the steady state errors and raise the time of the system. It is useful for changing the magnitude and adding phase lag to the output. It controls the system from fluctuating and also has the ability to return the system to its set point.

3.1.2 Disadvantages

It cannot handle systems with high-frequency response, It is also sensitive to controller gains. It results in steady state error if the system has a large disturbance.

3.2 Using a PD Controller

In a PD (Proportional-Derivative) controller, the controller uses two responses, namely the proportional and derivative terms, to generate the control signal that minimizes the error. Here's how you can express the equation for a PD controller

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

The technique which we can employ for controlling the double joint system with inputs f_1 and f_2 is to employ two independent controllers, one for each link. We can write the following equations:

$$
\begin{aligned}
f_1 &= K_{P_1} e_1(q_1) + K_{D_1} \dot{e}_1(q_1) \\
&= K_{P_1}(q_{1f} - q_1) - K_{D_1} \dot{q}_1
\end{aligned}
$$

$$
\begin{aligned}
f_2 &= K_{P_2} e_2(q_2) + K_{D_2} \dot{e}_2(q_2) \\
&= K_{P_2}(q_{2f} - q_2) - K_{D_2} \dot{q}_2
\end{aligned}
$$

The complete system of equations with control is then

$$
\ddot{q} = -M^{-1}(q)[c(q, \dot{q}) + G(q)] + \hat{F}
$$

$$
\hat{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} K_{P_1}(q_{1f} - q_1) - K_{D_1} \dot{q}_1 \\ K_{P_2}(q_{2f} - q_2) - K_{D_2} \dot{q}_2 \end{bmatrix}
$$

.

The actual physical torques are,

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

.

$$
\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = -M^{-1}(q)[c(q, \ddot{q}) + G(q)] + \begin{bmatrix} K_{P_1}(q_{1f} - q_1) - K_{D_1} \dot{q}_1 \\ K_{P_2}(q_{2f} - q_2) - K_{D_2} \dot{q}_2 \end{bmatrix} \tag{17}
$$

$$
\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} K_{P_1}(q_{1f} - q_1) - K_{D_1} \dot{q}_1 \\ K_{P_2}(q_{2f} - q_2) - K_{D_2} \dot{q}_2 \end{bmatrix} \tag{18}
$$

We now discretise the differential equations above by transforming them into a first-order ordinary differential equation system. This is accomplished by introducing new variables:

$$
u_1 = q_1,\ u_2 = q_2,\ u_3 = \dot{q}_1,\ u_4 = \dot{q}_2, \tag{19}
$$

$$
\dot{u_1} = \dot{q_1} = u_3
$$

$$
\dot{u_2} = \dot{q_2} = u_4
$$

$$\ddot{u_3} = \ddot{q_1} = \phi(t, u_1, u_2, u_3, u_3, u_4)$$
$$\ddot{u_4} = \ddot{q_2} = \psi(t, u_1, u_2, u_3, u_3, u_4)$$

where ϕ and ψ are expressed in terms of u:

$$\begin{bmatrix}\phi \\ \psi\end{bmatrix} = -M^{-1}(q)[c(q, \ddot{q}) + G(q)] + \begin{bmatrix}K_{P_1}(q_{1f} - u_1) - K_{D_1}u_3 \\ K_{P_2}(q_{2f} - u_2) - K_{D_2}u_4\end{bmatrix}$$

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

Now, we have a system of first-order non linear differential equations of the firm:

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

where,

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

and

$$H = \begin{bmatrix} u_3 \\ u_4 \\ \phi(t, u_1, u_2, u_3, u_3, u_4) \\ \psi(t, u_1, u_2, u_3, u_3, u_4)\end{bmatrix}$$

.

The initial conditions are

$$U_o = \begin{bmatrix}q_1(0) \\ q_2(0) \\ \dot{q_1}(0) \\ \dot{q_2}(0)\end{bmatrix}$$

This is solved using the ode45 command in Matlab(see code). This gives us the torque.

$$\begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = M(q) \begin{bmatrix}K_{P_1}(q_{1f} - u_1) - K_{D_1}u_3 \\ K_{P_2}(q_{2f} - u_2) - K_{D_2}u_4\end{bmatrix} \tag{20}$$

Equating and rearranging, we get,

$$\tau_1 = (M_1 + M_2)L_1^2(K_{P_1}(q_{1f} - u_1) - K_{D_1}u_3 +$$

$$+M_2L_1L_2\cos(u_1 - u_2)(K_{P_2}(q_{2f} - u_2) - K_{D_2}u_4,$$

$$\tau_2 = M_2L_1L_2\cos(u_1 - u_2))(K_{P_1}(q_{1f} - u_1) - K_{D_1}u_3$$
$$+M_2L_2^2(K_{P_2}(q_{2f} - u_2) - K_{D_2}u_4$$

.

3.2.1 Advantages

PD controller is used to reduce the transient errors like raise time, oscillations and overshoot of the system. It is useful for changing the magnitude and adding phase lead to the output. It is used in systems that must act very quickly, offering a response that causes the output to continually change in value.

3.2.2 Disadvantages

A disadvantage occurs when the input modifications are instantaneous, the variation speed will be very high, so the response of the differential regulator will be very abrupt.

3.3 Using a PID Controller

In a PID system, the controller employs all three responses (proportional, integral, and derivative) to generate the control signal that minimizes the error. The equation for a PID controller is generally expressed as:

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

The technique which we can employ for controlling the double joint system with inputs f_1 and f_2 is to employ two independent controllers, one for each link. We can write the following equations:

$$f_1 = K_{P_1} e_1(q_1) + K_{D_1} \dot{e}_1(q_1) + K_{I_1} \int e(q_1)\, dt$$
$$= K_{P_1}(q_{1f} - q_1) - K_{D_1} \dot{q}_1 + K_{I_1} \int (q_{1f} - q_1)\, dt$$

$$f_2 = K_{P_2} e_2(q_2) + K_{D_2} \dot{e}_2(q_2) + K_{I_2} \int e(q_2)\, dt$$
$$= K_{P_2}(q_{2f} - q_2) - K_{D_2} \dot{q}_2 + K_{I_2} \int (q_{2f} - q_2)\, dt$$

The complete system of equations with control is then

$$\ddot{q} = -M^{-1}(q)[c(q, \dot{q}) + G(q)] + \hat{F}$$

$$\hat{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} K_{P_1}(q_{1f} - q_1) - K_{D_1}\dot{q}_1 + K_{I_1} \int (q_{1f} - q_1)\, dt \\ K_{P_2}(q_{2f} - q_2) - K_{D_2}\dot{q}_2 + K_{I_2} \int (q_{2f} - q_2)\, dt \end{bmatrix}$$

.
The actual physical torques are,

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

.
To implement PID controller we introduce new states

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

differentiating with respect to t gives
$\dot{x}_1 = e(q_1) = q_{1f} - q_1$, $\dot{x}_2 = e(q_2) = q_{2f} - q_2$.

$$\dot{x_1} = q_{1f} - q_1 \tag{21}$$

$$\dot{x_2} = q_{2f} - q_2 \tag{22}$$

$$\begin{bmatrix}\ddot{q_1}\\ \ddot{q_2}\end{bmatrix} = -M^{-1}(q)[c(q,\ddot{q}) + G(q)] + \begin{bmatrix}K_{P_1}(q_{1f} - q_1) - K_{D_1}\dot{q_1} + K_{I_1}x_1\\ K_{P_2}(q_{2f} - q_2) - K_{D_2}\dot{q_2} + K_{I_2}x_2\end{bmatrix} \quad (23)$$

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = M(q)\begin{bmatrix}K_{P_1}(q_{1f} - q_1) - K_{D_1}\dot{q_1} + K_{I_1}x_1\\ K_{P_2}(q_{2f} - q_2) - K_{D_2}\dot{q_2} + K_{I_2}x_2\end{bmatrix} \quad (24)$$

We now discretise the differential equations above by transforming them into a first-order ordinary differential equation system. This is accomplished by introducing new variables:

$$u_1 = x_1,\ u_2 = x_2,\ u_3 = q_1,\ u_4 = q_4,\ u_5 = \dot{q_1},\ u_6 = \dot{q_2}, \quad (25)$$

Once differentiated,

$$\begin{aligned}\dot{u_1} = \dot{x_1} &= q_{1f} - u_3, \quad (26)\\ \dot{u_2} = \dot{x_2} &= q_{2f} - u_4,\\ \dot{u_3} = \dot{q_1} &= u_5\\ \dot{u_4} = \dot{q_2} &= u_6\\ \ddot{u_5} = \ddot{q_1} &= \phi(t, u_1, u_2, u_3, u_3, u_4, u_5, u_6)\\ \ddot{u_6} = \ddot{q_2} &= \psi(t, u_1, u_2, u_3, u_3, u_4, u_5, u_6)\end{aligned}$$

where ϕ and ψ are expressed in terms of u:

$$\begin{bmatrix}\phi\\ \psi\end{bmatrix} = -M^{-1}(q)[c(q,\ddot{q}) + G(q)] + \begin{bmatrix}K_{P_1}(q_{1f} - u_3) - K_{D_1}u_5 + K_{I_1}u_1\\ K_{P_2}(q_{2f} - u_4) - K_{D_2}u_6 + K_{I_2}u_2\end{bmatrix}$$

$$q = \begin{bmatrix}u_3\\ u_4\end{bmatrix}$$

Now, we have a system of first-order non-linear differential equations of the firm:

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

where,

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

and

$$H = \begin{bmatrix} q_{1f} - u_3 \\ q_{2f} - u_4 \\ u_5 \\ u_6 \\ \phi(t, u_1, u_2, u_3, u_3, u_4, u_5, u_6) \\ \psi(t, u_1, u_2, u_3, u_3, u_4, u_5, u_6) \end{bmatrix}$$

.

The initial conditions are

$$U_0 = \begin{bmatrix} x_1(0) \\ q_2(0) \\ q_1(0) \\ q_2(0) \\ \dot{q_1}(0) \\ \dot{q_2}(0) \end{bmatrix}$$

This is solved using the ode45 command in Matlab(see code). This gives us the torque.

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} K_{P_1}(q_{1f} - u_3) - K_{D_1}u_5 + K_{I_1}u_1 \\ K_{P_2}(q_{2f} - u_4) - K_{D_2}u_6 + K_{I_2}u_2 \end{bmatrix} \tag{27}$$

Equating and rearranging, we get,

$$\tau_1 = (M_1 + M_2)L_1^2(K_{P_1}(q_{1f} - u_3) - K_{D_1}u_5 + K_{I_1}u_1)$$
$$+M_2L_1L_2\cos(u_3 - u_4)(K_{P_2}(q_{2f} - u_4) - K_{D_2}u_6 + K_{I_2}u_2),$$

$$\tau_2 = M_2L_1L_2\cos(u_3 - u_4))(K_{P_1}(q_{1f} - u_3) - K_{D_1}u_5 + K_{I_1}u_1)$$
$$+M_2L_2^2(K_{P_2}(q_{2f} - u_4) - K_{D_2}u_6 + K_{I_2}u_2$$

We can compute these equations to solve them.

3.3.1 Advantages

PID controller is a general form of controller. The gains of all 3 control actions are adjusted to achieve this controller. Adjusting these gains, any combination of P, I, and D controllers can be obtained which makes this a more robust one. It is useful for changing the magnitude along with either lag or lead in the phase to the output.

3.3.2 Disadvantages

One of the main disadvantages of PID controllers is that they can be sensitive to noise and measurement errors, as they can amplify the fluctuations in the input signal and cause instability or oscillations. PID controllers may also face difficulties with non-linear systems, as they can cause performance degradation or instability when the system dynamics change significantly or unpredictably.

4 Implementing in Matlab

4.1 Matlab Code

```
%Given Constants
m1=10;
m2=5;
l1=0.2;
l2=0.1;
g=9.81;

%Initial State Variables
q1=0;
q2=0;
q3=0.1;
q4=0.1;
q3_dot = 0;
q4_dot = 0;
init_q=[\(q_1\),\(q_2\), q3, q4, q3_dot, q4_dot];

%PI controller gains
kp=100;
Ki=0.5;
gain=[kp;Ki];

%Final Angles
qf=[0;0];

options = odeset('RelTol', 1e-3, 'AbsTol', 1e-6);
[time,s] = ode45(@(t,q) pid(t,q,qf,gain), [0, 30],
   init_q,options);

q_final=zeros(length(time),1);

figure;
subplot(2,1,1);
plot(time,s(:,3));
axis([0,30,-1,1]);
xlabel("Time");
ylabel("q_1");
```

```
hold on;
plot(time,q_final,'g');
hold off;

subplot(2,1,2);
plot(time,s(:,4),'r');
axis([0,30,-1,1]);
xlabel("Time");
ylabel("q_2");
hold on;
plot(time,q_final,'g');
hold off;

sgtitle("Joint Angle vs Time - PI Controller");

function [q_double] = pid(t,q,qf,gains)
    %Given Constants
    m1=10;
    m2=5;
    l1=0.2;
    l2=0.1;
    g=9.81;

   %Error
    error = [qf(1) - q(3);
         qf(2) - q(4)];

    f1 = gains(1) * error(1) + gains(2)*q(1);
    f2 = gains(1) * error(2) + gains(2)*q(2);
    f = [f1; f2];

    %Mass Matrix
    M11 = (m1 + m2) * (l1^2) + m2 * l2*(l2 + 2 * l1 *
       cos(q(4)));
    M22 = m2 * l2^2;
    M12 = m2*l2*(l2+l1*cos(q(4)));
    M21 = M12;
    M = [M11, M12; M21, M22];

    %Coriolis Matrix
```

```
    c11 = -m2 * l1 * l2 * sin(q(4)) * q(5);
    c12 = -m2 * l1 * l2 * sin(q(4)) * (q(5) + q(6));
    c21 = 0;
    c22 = m2 * l1 * l2 * sin(q(4)) * q(6);
    C = [c11, c12; c21, c22];

    %Gravitational Matrix
    G1 = m1*l1*g*cos(q(3)) + m2*g*(l2*cos(q(3)+q(4)) +
       l1*cos(q(3)));
    G2 = m2 * l2 * g * cos(q(3)+q(4));
    G = [G1;G2];

    dq = [q(5); q(6)];

    m_inverse = inv(M);
    W = ((m_inverse) * (-C * dq - G)) + f;

    %Next state
    q_double=zeros(4,1);
    q_double(1)=qf(1)-q(3);
    q_double(2)=qf(2)-q(4);
    q_double(3)=q(5);
    q_double(4)=q(6);
    q_double=[q_double;W];
end
```

Listing 1: PI

```
%Given Constants
m1=10;
m2=5;
l1=0.2;
l2=0.1;
g=9.81;

%Initial State Variables
q1=0.1;
q2=0.1;
q1_dot = 0;
q2_dot = 0;
init_q=[q1,q2, q1_dot, q2_dot];

%PD controller Gains
kp=100;
kd=50;
gain=[kp;kd];

%Final Angles
qf=[0;0];

options = odeset('RelTol', 1e-3, 'AbsTol', 1e-6);
[time,s] = ode45(@(t,q) pd(t,q,qf,gain), [0, 30],
   init_q,options);

figure;
subplot(2,1,1);
plot(time,s(:,1));
xlabel("Time");
ylabel("q_1");

subplot(2,1,2);
plot(time,s(:,2),'r');
xlabel("Time");
ylabel("q_2");

sgtitle("Joint Angle vs Time - PD Controller");

```

```
function [q_double] = pd(t,q,qf,gains)
    %Given Constants
    m1=10;
    m2=5;
    l1=0.2;
    l2=0.1;
    g=9.81;

    %Error
    error = [qf(1) - q(1);
         qf(2) - q(2)];

    f1 = gains(1) * error(1) - gains(2) * q(3);
    f2 = gains(1) * error(2) - gains(2) * q(4);


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

    %Coriolis Matrix
    c11 = -m2 * l1 * l2 * sin(q(2)) * q(3);
    c12 = -m2 * l1 * l2 * sin(q(2)) * (q(3) + q(4));
    c21 = 0;
    c22 = m2 * l1 * l2 * sin(q(2)) * q(4);
    C = [c11, c12; c21, c22];

    %Gravitational Matrix
    G1 = m1*l1*g*cos(q(1)) + m2*g*(l2*cos(q(1)+q(2)) +
       l1*cos(q(1)));
    G2 = m2 * l2 * g * cos(q(1)+q(2));
    G = [G1;G2];

    f = [f1; f2];
    dq = [q(3); q(4)];

```

```
77      m_inverse = inv(M);
78      W = ((m_inverse) * (-C * dq - G)) + f;
79
80      %Next state
81      q_double=zeros(2,1);
82      q_double(1)=q(3);
83      q_double(2)=q(4);
84      q_double=[q_double;W];
85  end
```

Listing 2: PD

```
% Given Constants
m1=10;
m2=5;
l1=0.2;
l2=0.1;
g=9.81;

% Initial State Variables
q1=0;
q2=0;
q3=0.1;
q4=0.1;
q3_dot = 0;
q4_dot = 0;
init_q=[q1,q2, q3, q4, q3_dot, q4_dot];

% PID Controller gains
kp=50;
kd=50;
ki=50;
gain=[kp;kd;ki];

% Final angles
qf=[0;0];

options = odeset('RelTol', 1e-3, 'AbsTol', 1e-6);
[time,s] = ode45(@(t,q) pid(t,q,qf,gain), [0, 30],
   init_q,options);

figure;
subplot(2,1,1);
plot(time,s(:,3));
xlabel("Time");
ylabel("q_1");

subplot(2,1,2);
plot(time,s(:,4),'r');
xlabel("Time");
ylabel("q_2");
```

```
39
40 sgtitle("Joint Angle vs Time - PID Controller");
41
42 function [q_double] = pid(t,q,qf,gains)
43     % ... (rest of your MATLAB code here)
44 end
```

Listing 3: PID

4.2 Simulink

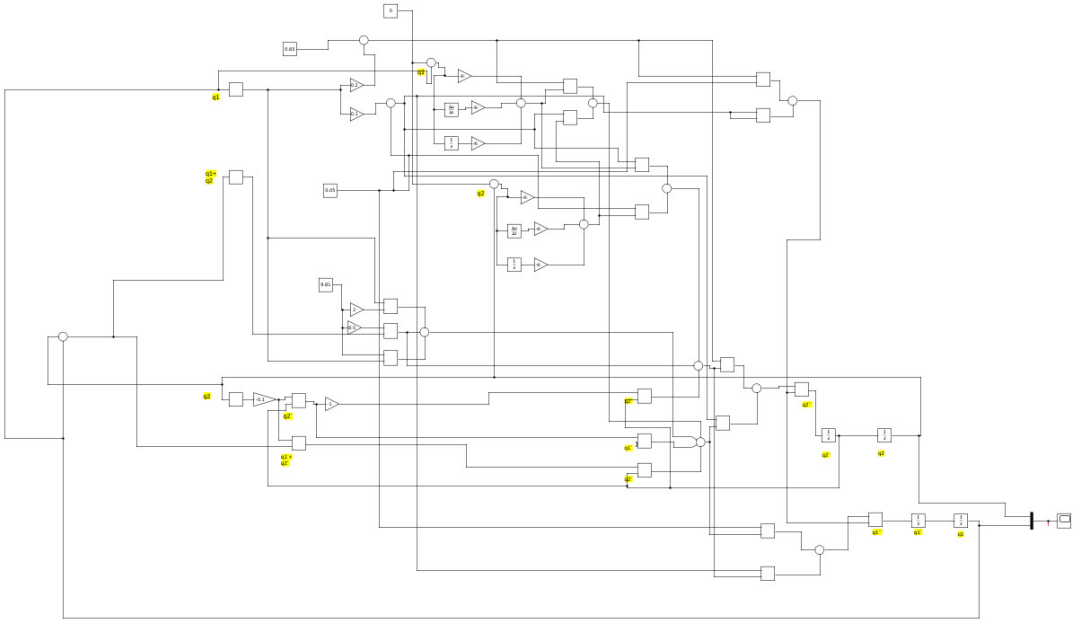


Figure 1: Simulink Diagram

4.3 Observations

In the below Simulink plots, the blue plot represents q2 and the yellow plot represents q1.

4.3.1 PI controller

We take the following values: $K_p = 100$, $K_i = 0.5$.

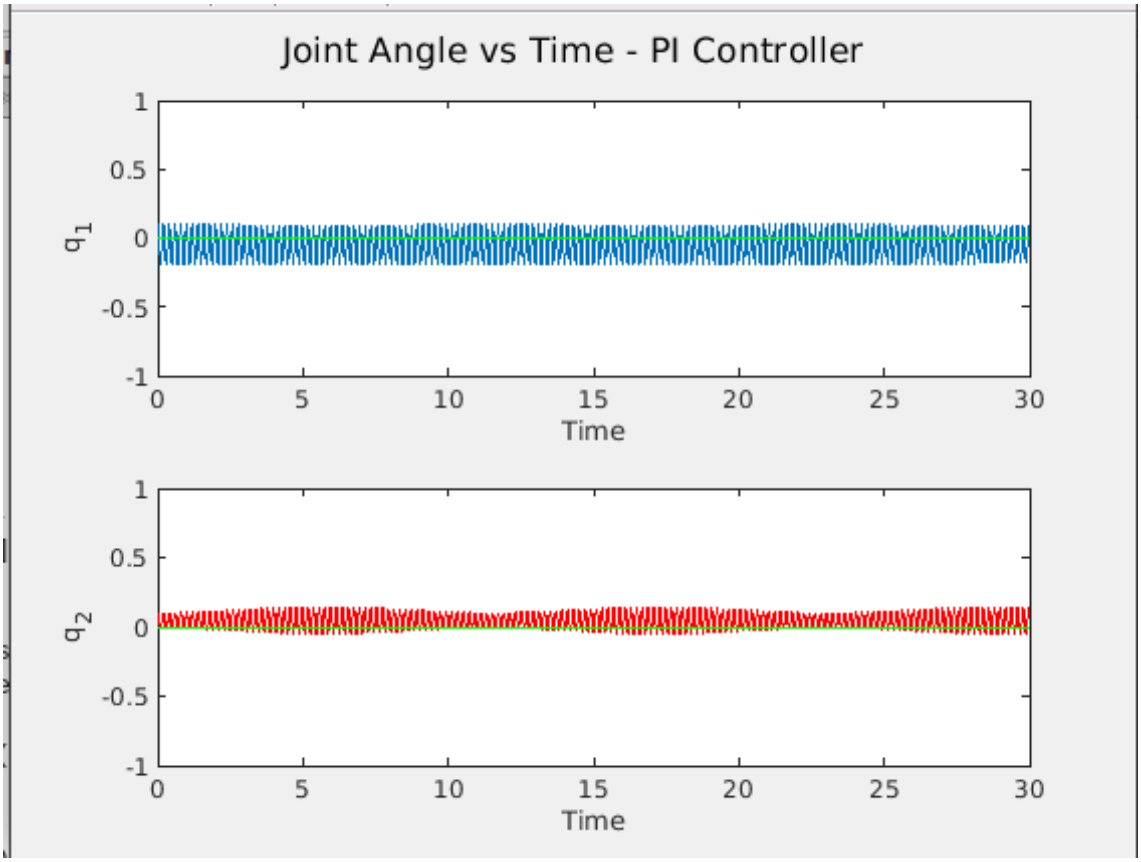


Figure 2: PI controller

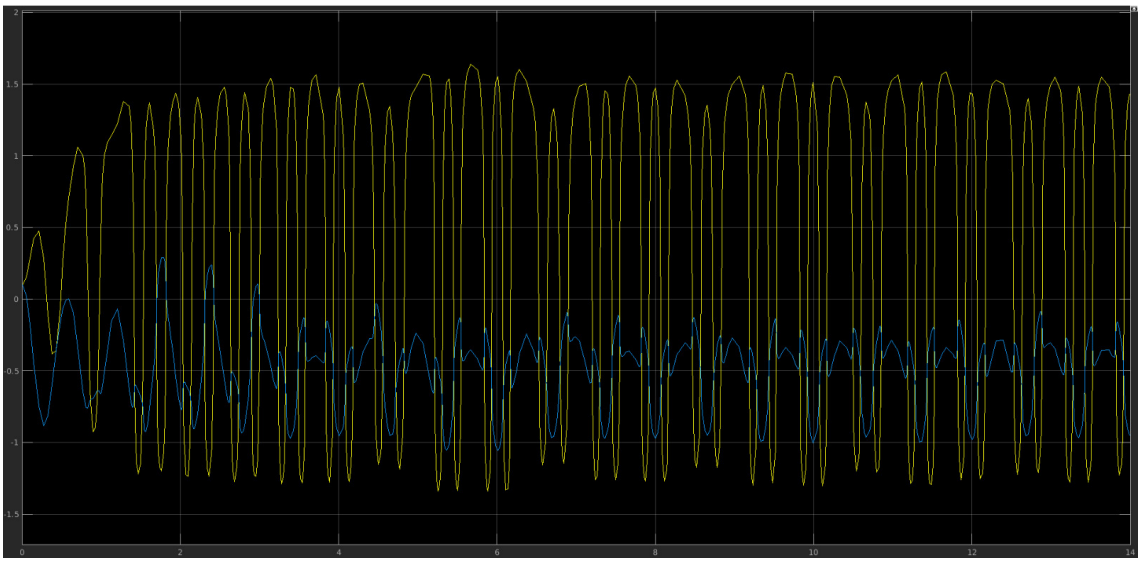


Figure 3: PI controller-simulink

We observe exaggerated oscillations due to the lack of a derivative term. The Joint angles of q_1 and q_2 did not achieve a steady state at 0 as desired. The error occurred as there was nothing correcting the overshoot observed in the signal.

Even if there isn't an error at a given moment, the integrator may still be impacted by past errors as the integrator has a memory of all the errors in the past. Because of this, it keeps correcting for a little longer until it overshoots or undershoots. This leads to the oscillations observed in the graph.

However, the steady-state error is less when compared to other systems, like the PD system due to the integral term which adds up the error over time and reduces the steady-state errors.

We take $K_p = 1000$, $K_i = 0.5$.

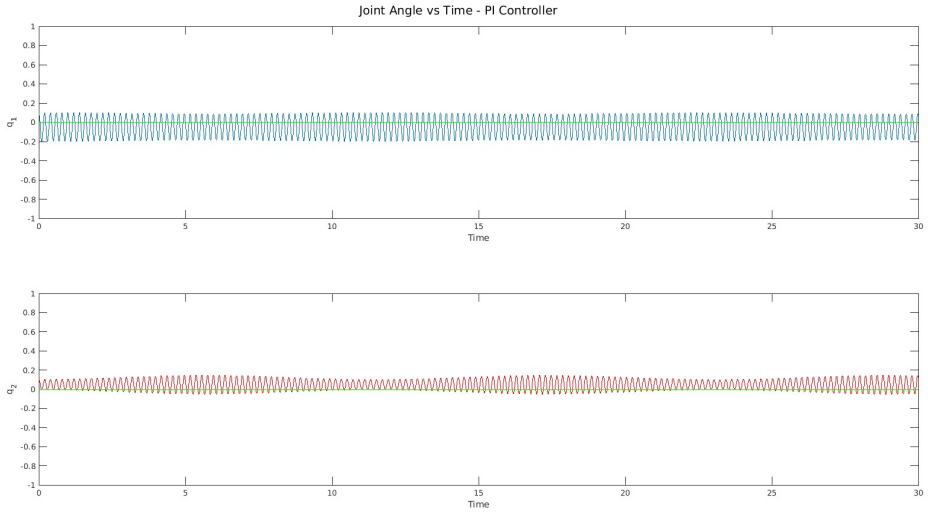


Figure 4: PI controller

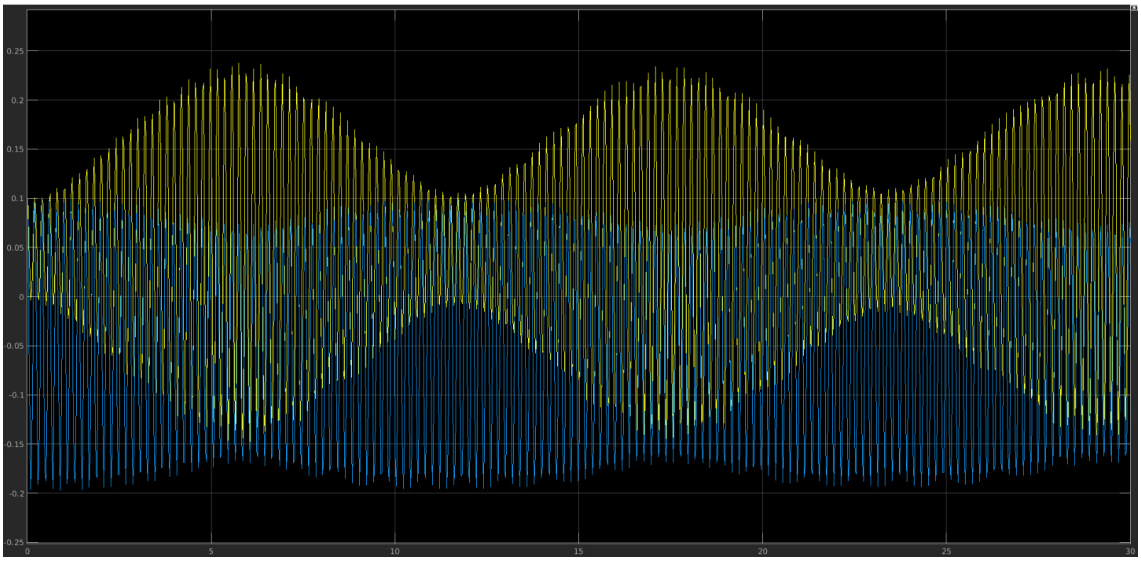


Figure 5: PI controller-simulink

In the above plot, we see that the oscillating error increases due to increase in the value of K_p. The integrated error remains the same due the

same value of K_i.

We take $K_p = 750$, $K_i = 1$.

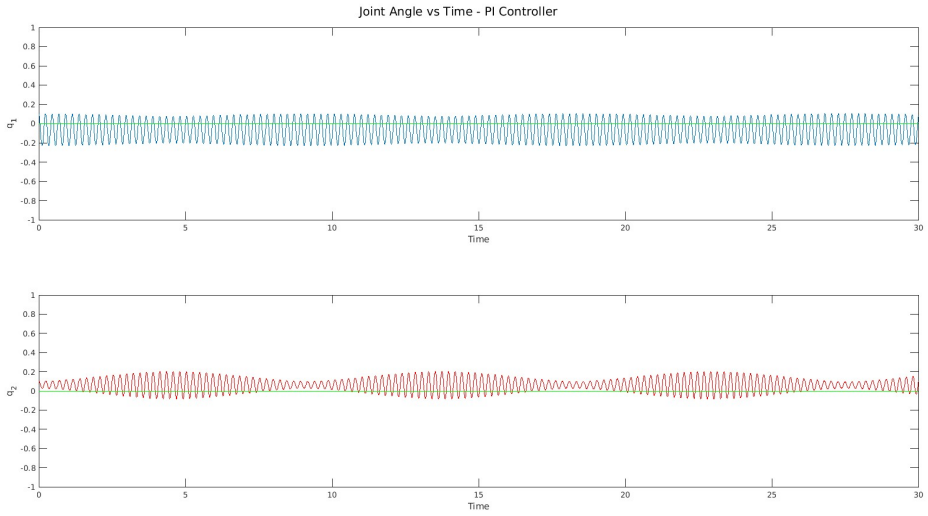


Figure 6: PI controller

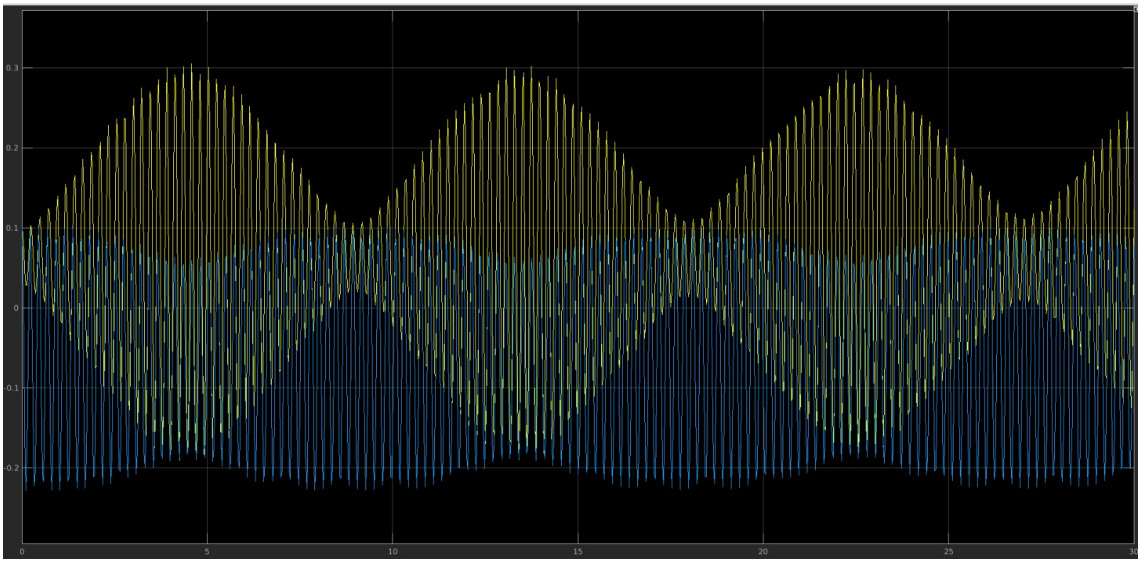


Figure 7: PI controller-simulink

We see a similar variance in the above plot since both K_p and K_i are changed. The effect of change in K_i is observed in the faster rate of oscillations of the error in the final plot.

4.3.2 PD controller

We take $K_p = 100$, $K_d = 50$:

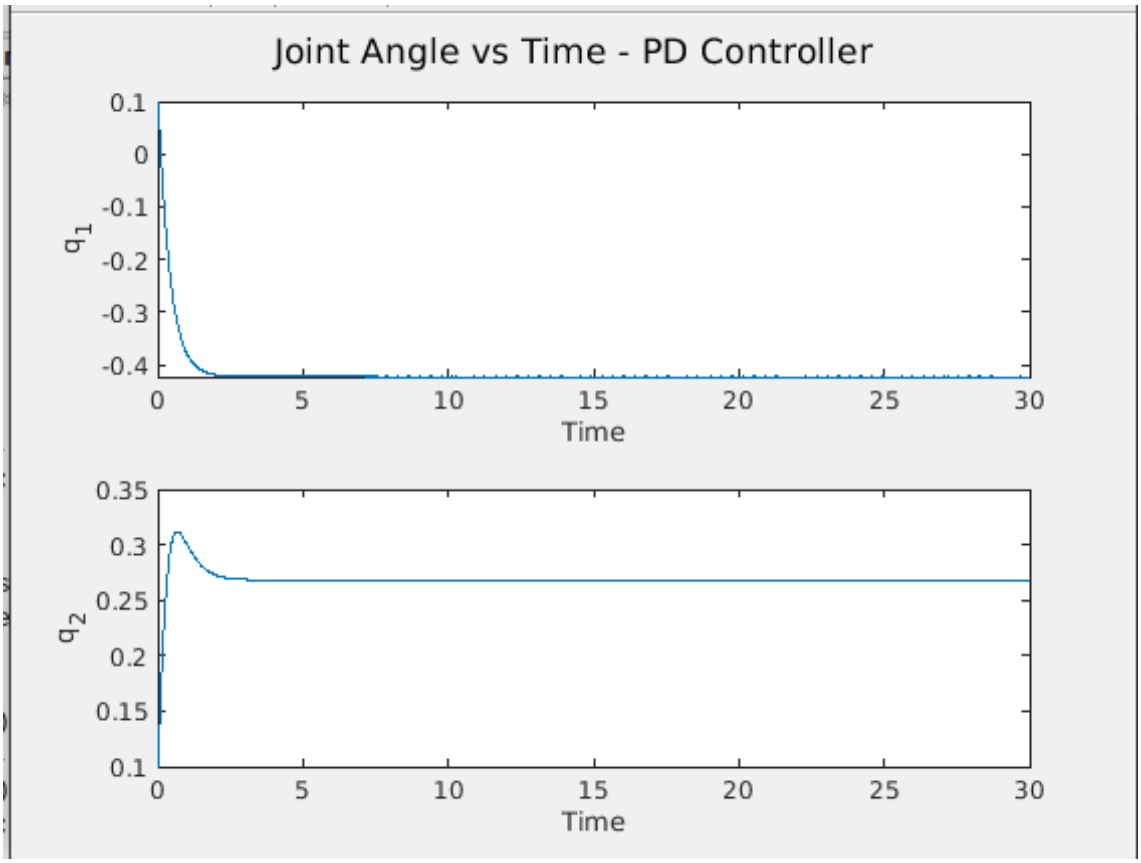


Figure 8: PD controller

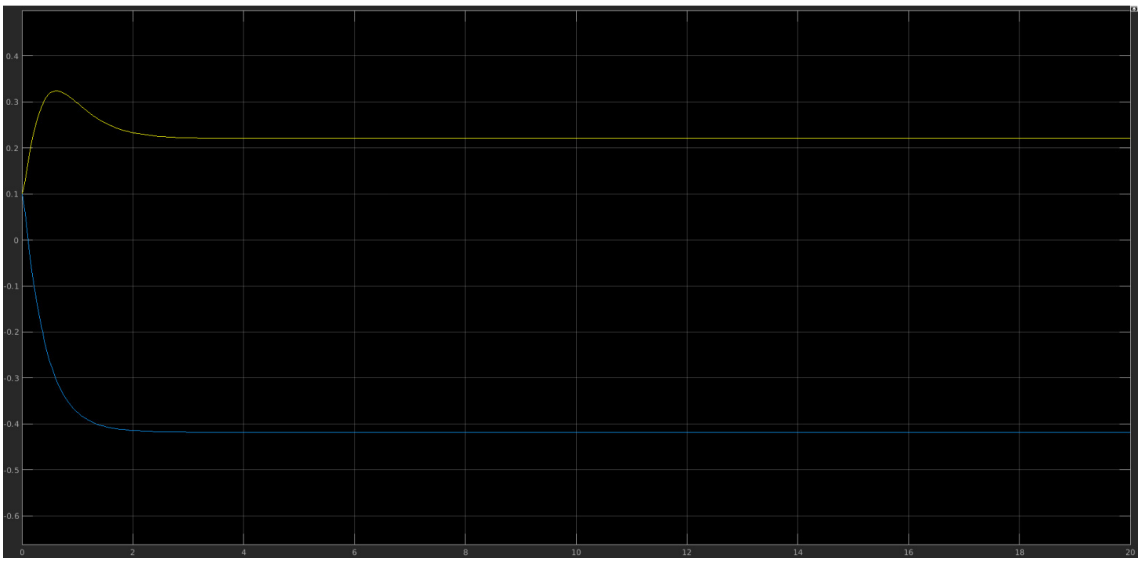


Figure 9: PD controller-simulink

We don't observe any exaggerated oscillation as observed in the PI case. The PD controller helped in correcting the overshoot and decreasing the sudden changes in the system and improves the response time. K_d acts as a damping factor.

The joint angles of q_1 and q_2 did not achieve a steady state at 0 as desired. The error occurred as there was nothing to correct the steady-state errors. We observe a steady state error which is slightly away from the desired output and is usually corrected by integral terms.

We take $K_p = 500$, $K_d = 100$:

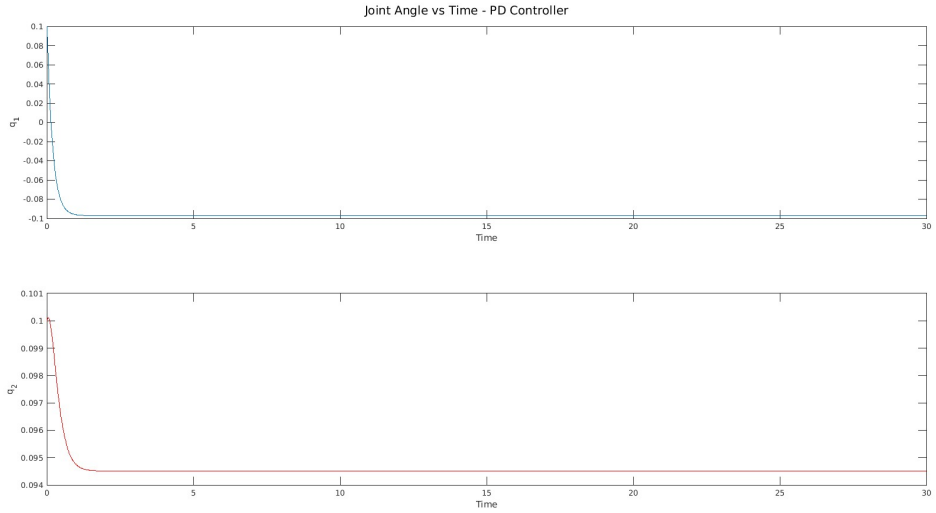


Figure 10: PD controller

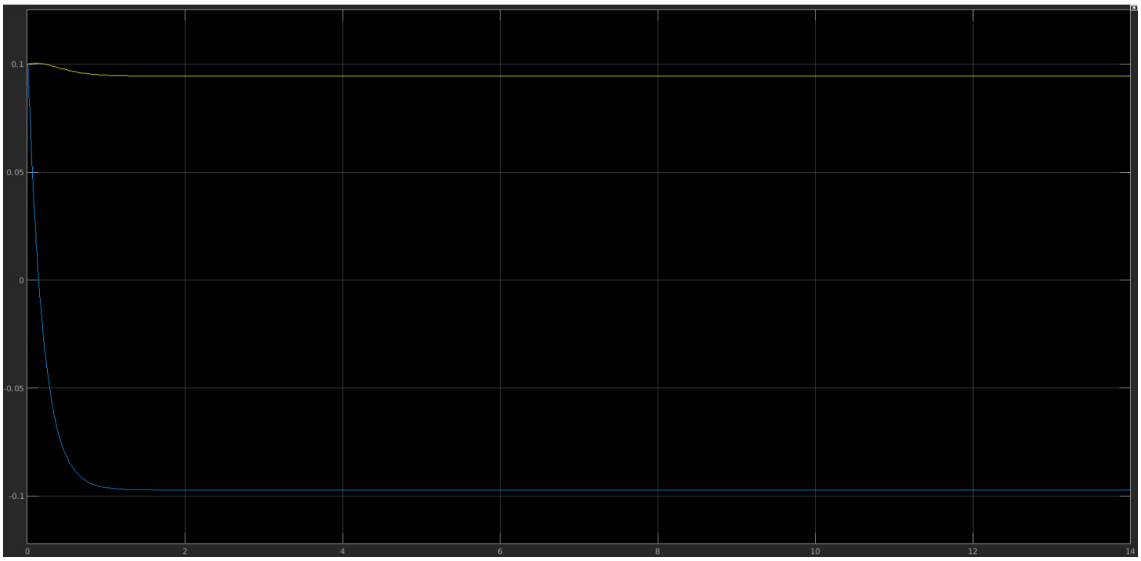


Figure 11: PD controller-simulink

Increasing K_p results in a higher error from the desired output of the controller. Increasing K_d, we observe that the overshoot of the signal decreases

which can observed in the above plot as compared to the first plot.

We take $K_p = 100$, $K_d = 100$:

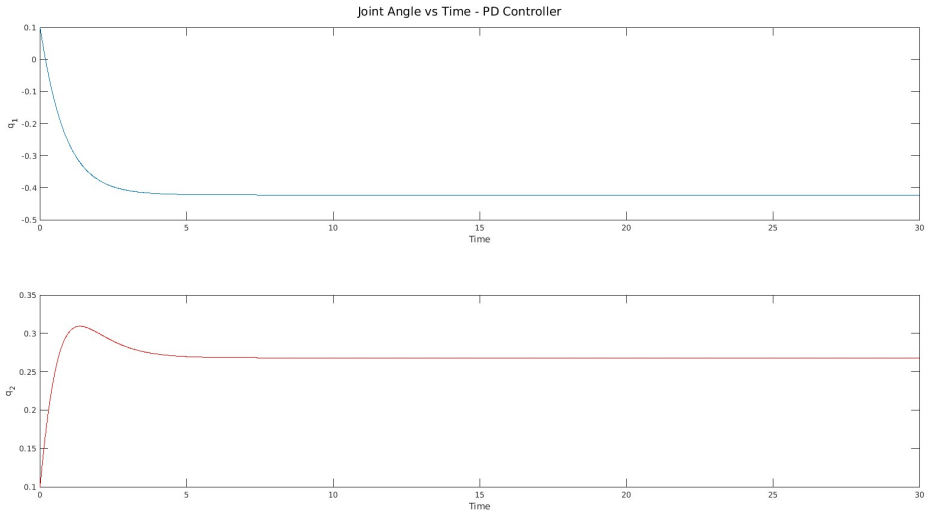


Figure 12: PD controller

Figure 13: PD controller-simulink

The same trend can be observed for the above plot as well.

4.3.3 PID controller

We take $K_p = 50$, $K_d = 50$, K_i=50:

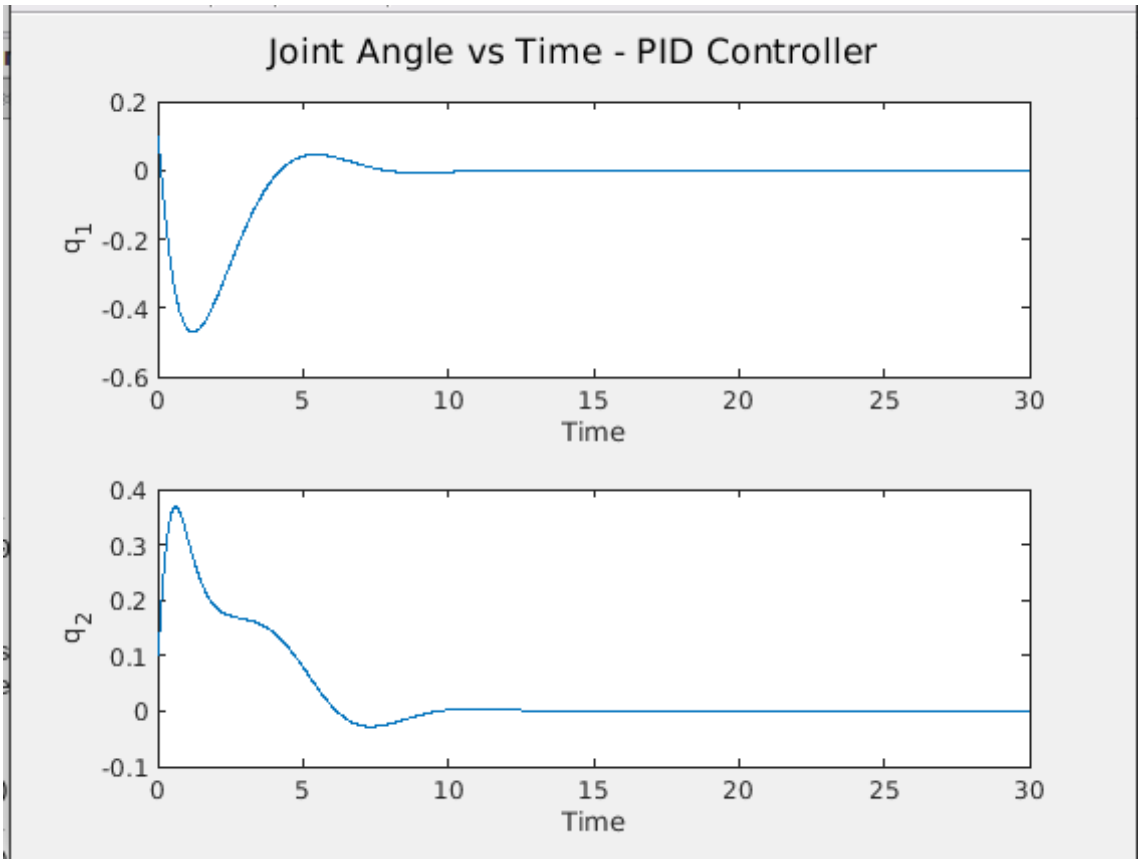


Figure 14: PID controller

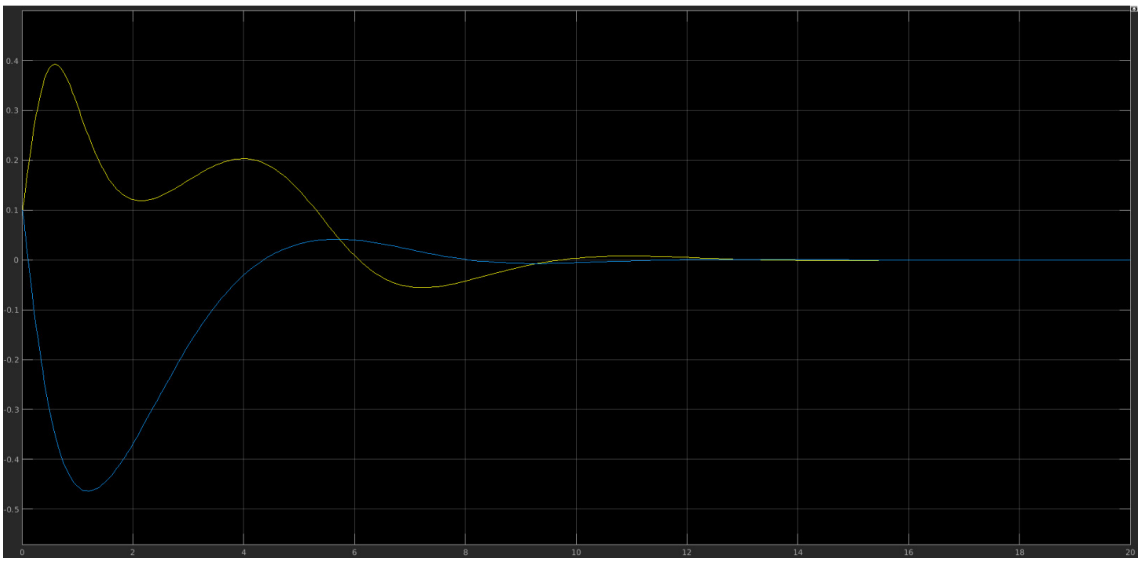


Figure 15: PID controller-simulink

All the properties of the PID system make the system achieve the desired result. The System reaches the state (0,0). The PID has the least amount of error and settling when compared to PI and PD controllers.

We observe that the joint angle in q_1 falls sharply. The PD part of the controller Decreases overshoot and settling time. while no steady-state error is observed due to the PI part of the controller.

For q_2 We observe that the joint angle rises suddenly then falls sharply and starts to reach a steady state. The PD part of the controller decreases the overshoot between 0 and 5. The rise time also decreases. The PI part of the controller decreases the steady state error in the system.

The proportional terms must be optimum. A higher P will give a snappier response, but the chances of overshooting increase and a lot of unwanted oscillations are generated. While a lower P will give a sluggish response. Which may lead it to never reach the steady state. The proportional term is selected carefully to avoid this error.

The derivative term helps in limiting the response of the system. This helps when the error is changing rapidly It helps in reducing the overshoot. Too much of this will make the system sluggish again. The correct K_d helped us reduce the discrepancies and oscillations seen in a PI controller.

The integral term adds up the error over time and reduces the steady-state

errors. A higher K_i is also not desirable as the possibility of overshooting increases ma$K_i$ng it unstable. Hence after the accurate value of K_i is taken the result is more accurate when compared to a PD controller.

Hence the K_p, K_i, k_d, are necessary to achieve the desired result

We take $K_p = 100$, $K_d = 100$, K_i=100:

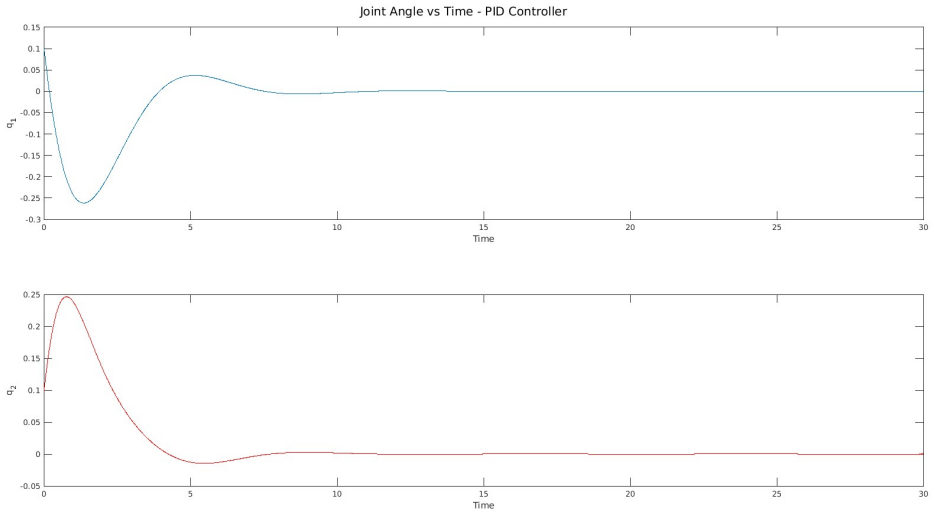


Figure 16: PID controller

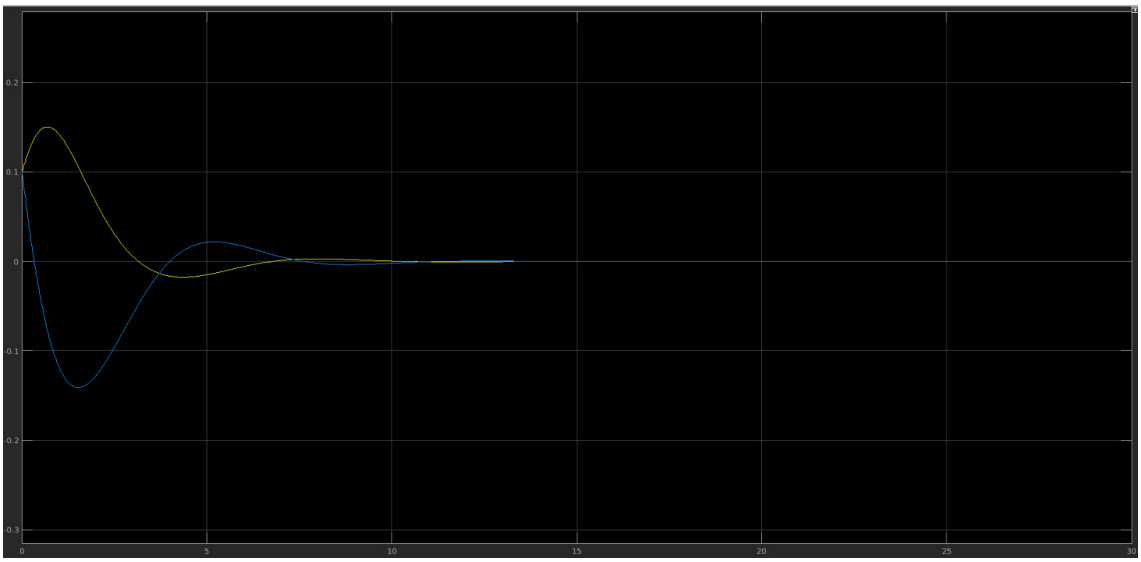


Figure 17: PID controller-simulink

Increasing K_i decreases the minute error in the final output; increasing K_d decreases the overshoot of the overall plot and the rise in K_p tries to increase the initial error which is balanced out by the increased value of K_d.

We take $K_p = 92$, $K_d = 92$, K_i=92:

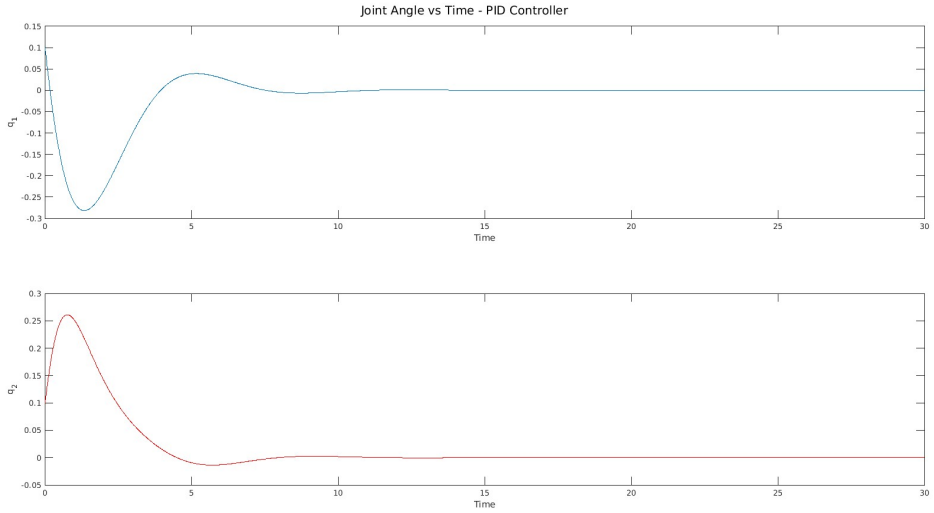


Figure 18: PID controller

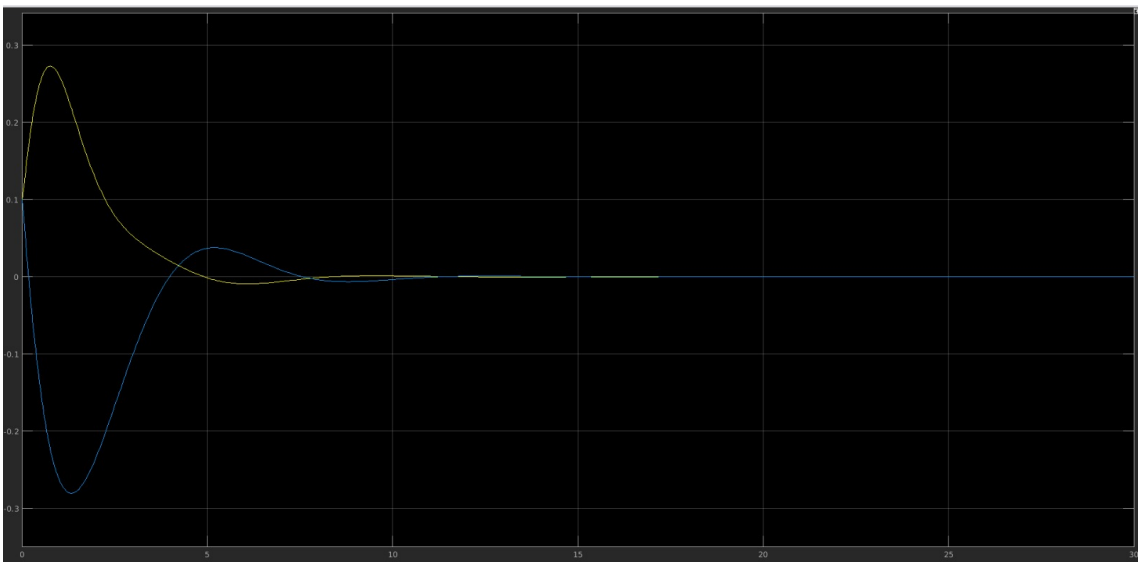


Figure 19: PID controller-simulink

Similar trend is observed for the above plots as well.

5 Conclusions

Two link manipulators have widespread applications and constitute a basic building block in robotics. In this project, we have focused on understanding the dynamics and control of the system using mathematical tools. Additionally, we have attempted to extrapolate our understanding by implementing the system in a simulated environment using Matlab-SIMULINK. As outlined in the project, control can be triggered with different variations of the proportional, derivative and integral responses. Drawing from our results, we were able to conclude that the use of the PID controller was the most effective and efficient. It should be noted however that the experiments were carried out in a synthetic environment, and the output might differ in the real world. That being said, the implications of our findings have far-reaching consequences and a plethora of research revolving around the same continues to this very day.