INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

PROJECT REPORT

SYSTEMS THINKING

By

THE ELITE EIGHT

SAUMYA BALINA
2022102069

HIMANI SHARMA
2022102032

MEEMANSA PANDEY
2022102036

KRIPI SINGLA
2022102063

JANYA GUPTA
2022102033

YASH SERI
2022102015

ZAINAB RAZA
2022102013

MOHAMMED NOOR
2021102014

CONTENTS

Chapter 1

DYNAMICS OF TWO-LINK MANIPULATOR

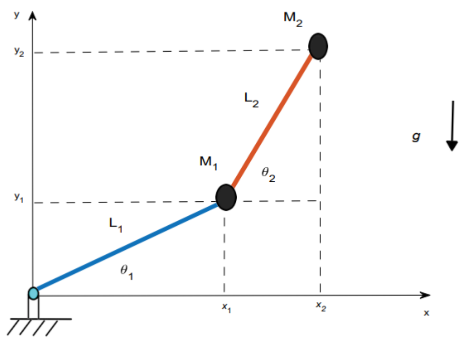


Figure 1: A simplified model of a two-link planar robot manipulator.

The system consists of two masses connected by weightless bars. 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 bar rotates about the origin and the second bar rotates about the endpoint of the first bar, respectively. This system has two degrees of freedom θ_1 and θ_2.

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.

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

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

$$y_1 = L_1 \sin(\theta_2)$$

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

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

$$y_2 = L_1\sin(\theta_1) + L_2\sin(\theta_2)$$

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

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

where,

$\dot{x}_1$ = -$L_1\dot{\theta}_1\sin(\theta_1)$, $\dot{y}_1$ = -$L_1\dot{\theta}_1\cos(\theta_1)$

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

Here and below the dot $\dot{}$ is a derivative with respect to t, i.e. $\dot{\theta}_k = \frac{d\theta_k}{dt}$, $\dot{x}_k = \frac{dx_k}{dt}$ for k=1,2.

The kinematic energy can be calculated as follows:

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

The equation for the kinetic energy can be written as:

KE = $\frac{1}{2}M_1(\dot{x}_1^2 + \dot{y}_1^2)$ + $\frac{1}{2}M_2(\dot{x}_1^2 + \dot{y}_1^2)$. which is simplified as:

KE = $\frac{1}{2}(M_1 + M_2){L_1}^2\dot{\theta_1}^2$ + $\frac{1}{2}M_2{L_2}^2\dot{\theta_2}^2$ + $M_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)$. In order to calculate the Lagrangian, the potential energy PE has to be calculated. By definition the potential energy of the system due to gravity of the i^{th} pendulum is:

PE_i = M_ig$h_i(\theta)$, i=1,2,

where h_i is the height of the center of mass of the i^{th} pendulum, g is the acceleration due to gravity constant, and M_i is the mass of i^{th} pendulum. Therefore, the total potential energy for both pendulum. Therefore, the total potential energy for both pendulums can be given by:

PE = $M_1gL_1\sin(\theta_1) + M_2g(L_1\sin(\theta_1) + L_2\sin(\theta_2)) = (M_1 + M_2)gL_1\sin(\theta_1) + M_2gL_2\sin(\theta_2)$

Next, by Lagrange Dynamics, we form the Lagrangian $\mathcal{L}$ which is defined as:

$\mathcal{L}$ = KE - PE

Substituting the expressions for the kinetic energy and potential energy in for KE and PE we get:

$$\begin{gathered}\mathcal{L} = \frac{1}{2}(M_1 + M_2){L_1}^2{\dot{\theta_1}}^2 + \frac{1}{2}M_2{L_2}^2{\dot{\theta_2}}^2 + M_2L_1L_2\dot{\theta_1}\dot{\theta_2}\cos(\theta_1 - \theta_2) \\ -(M_1 + M_2)gL_1\sin(\theta_1) - M_2gL_2\sin(\theta_2)\end{gathered}$$

The Euler-Lagrange equation is given by the equation:

$$\begin{aligned}\frac{d(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_k})}{dt} - \frac{\partial\mathcal{L}}{\partial\theta_1} &= \tau_i, i = 1, 2 \\ \frac{\partial\mathcal{L}}{\partial\dot{\theta_1}} &= (M_1 + M_2){L_1}^2\dot{\theta_1} + M_2L_1L_2\dot{\theta_2}(\cos(\theta_1 - \theta_2)) \\ \frac{\partial\mathcal{L}}{\partial\theta_1} &= -M_2L_1L_2\dot{\theta_1}\dot{\theta_2}(\sin(\theta_1 - \theta_2) - (M_1 + M_2)gL_1\cos(\theta_1) \\ \frac{d(\frac{\partial\mathcal{L}}{\partial\dot{\theta_1}})}{dt} &= (M_1 + M_2){L_1}^2\ddot{\theta_1} + M_2L_1L_2\ddot{\theta_2}\cos(\theta_1 - \theta_2) - M_2L_1L_2\dot{\theta_2}(\dot{\theta_1} - \dot{\theta_2})\sin(\theta_1 - \theta_2)\end{aligned}$$

Similarly, we compute

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\dot{\theta_2}} &= M_2{L_1}^2\dot{\theta_2} + M_2L_1L_2\dot{\theta_1}(\cos(\theta_1 - \theta_2)) \\ \frac{\partial\mathcal{L}}{\partial\theta_2} &= -M_2L_1L_2\dot{\theta_1}\dot{\theta_2}(\sin(\theta_1 - \theta_2) + M_2gL_2\cos(\theta_2) \\ \frac{d(\frac{\partial\mathcal{L}}{\partial\dot{\theta_2}})}{dt} &= M_2{L_2}^2\ddot{\theta_2} + M_2L_1L_2\ddot{\theta_1}\cos(\theta_1 - \theta_2) - M_2L_1L_2\dot{\theta_1}(\dot{\theta_1} - \dot{\theta_2})\sin(\theta_1 - \theta_2)\end{aligned}$$

Therefore, the following are two nonlinear equations of motion which are second-order system of ordinary differential equations:

$$\begin{gathered}(M_1 + M_2){L_2}^2\ddot{\theta_1} + M_2L_1L_2\ddot{_2}\cos\theta_1 - \theta_2 + M_2L_1L_2{\dot{\theta_2}}^2\sin\theta_1 - \theta_2 + (M_1 + M_2)gL_1\cos\theta_1 = \tau_1 \\ M_2L_2\ddot{\theta_2} + M_2l_1l_2\ddot{\theta_2}\cos(\theta_1 - \theta_2) - M_2L_1L_2{\dot{\theta_1}}^2\sin(\theta_1 - \theta_2) + M_2gL_2\cos(\theta_2) = \tau_2\end{gathered}$$

or equivalently,

$$L_1\ddot{\theta}_1 + \delta L_1 L_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) = \frac{\delta \tau_1}{M_2 L_1} - \delta L_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - g\cos\theta_1,$$
$$L_2\ddot{\theta}_2 + L_1 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) = \frac{\tau_2}{M_2 L_2} - L_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - g\cos\theta_2,$$

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 equation:

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

where,

$$g_1 = \frac{\frac{\delta\tau_1}{M_2L_1} - \delta L_2 \dot{\theta}_2^{\,2} \sin\theta_1 - \theta_2 - g\cos\theta_1 - \delta\cos(\theta_1 - \theta_2)(\frac{\tau_2}{M_2L_2} + L_1\dot{\theta}_1^{\,2}\sin\theta_1 - \theta_2 - g\cos\theta_2)}{L_1(1 - \delta(\cos(\theta_1 - \theta_2))^2)}$$
$$g_2 = \frac{\frac{\delta\tau_2}{M_2L_2} - L_1 \dot{\theta}_1^{\,2} \sin\theta_1 - \theta_2 - g\cos\theta_2 - \cos(\theta_1 - \theta_2)(\frac{\delta\tau_1}{M_2L_1} + \delta L_2\dot{\theta}_2^{\,2}\sin\theta_1 - \theta_2 - g\cos\theta_1)}{L_2(1 - \delta(\cos(\theta_1 - \theta_2))^2)}$$

In order to solve for the angles θ_1 and θ_2 , we need to solve the above second-order system of ordinary differential equations. To do this, we first reduce the system into an equivalent system of first-order ordinary differential equations.

Let us introduce four new variables:

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

After differentiating, we have

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

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

$\frac{d\boldsymbol{U}}{dt} = \boldsymbol{S}(t, \boldsymbol{U}), \boldsymbol{U}(0) = \boldsymbol{U_0}$,

$where, \boldsymbol{U}$ =[u$_1, u_2, u_3, u_4]^t$ and $\boldsymbol{S}$ =[s$_1, s_2, s_3, s_4]^t$ with

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

The initial conditions are given by:

$\boldsymbol{U_0}$ =[u$_1(0), u_2(0), u_3(0), u_4(0)]^t$,

where,

$u_1(0) = \theta_1(0), u_2(0) = \theta_2(0), u_3(0) = \dot{\theta}_3(0), u_4(0) = \dot{\theta}_4(0)$.

We are taking the initial angles as:

$\theta_1 = 0.1$

$\theta_2 = 0.1$

$\dot{\theta}_1 = 0$

$\dot{\theta}_2 = 0$

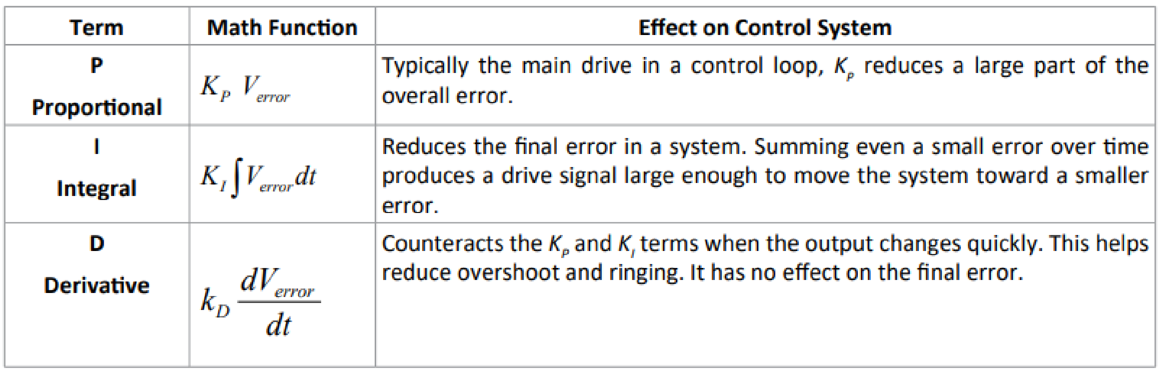
This system subject to the initial conditions can be solved for the unknown vector U ,using the ode45 command defined in MATLAB to solve the system of ordinary differential equations numerically.

Chapter 2

MODEL OF ROBOTIC MANIPULATOR

Robotic manipulators are generally difficult to control. In particular, it is a challenging task to stabilize a robot manipulator at a fixed, accurate position. In this section, we focus mainly on control of the robot manipulator to get the desired position using computed torque control method. After deriving the equation of motion, control simulation is represented using MATLAB.

Here we will be building algorithm for PD, PI and PID Controller Design. The "P" stands for Proportional control, the "I" stands for Integral control, and the "D" stands for Derivative control. The algorithm used works by defining an error variable $V_error = V_set$ V_sensor that takes the position we want to go (V_set) minus the position we are actually at (V_sensor). We get the proportional part of the PID control by taking a constant defined as K_P and multiplying it by the error. The I comes from taking a constant K_I and multiplying it by the integral of the error with respect to time. Derivative control is defined as a constant K_D multiplied by the derivative of the error with respect to time. Below is a table describing PID control:



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

The equations of motion can be written compactly as:

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

where,

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

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

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

$$\mathrm{C} = \begin{bmatrix} -m_2 l_1 l_2 sin(q_2)\dot{q_2} & -m_2 l_1 l_2 sin(q_2)(\dot{q_1} + \dot{q_2}) \\ 0 & m_2 l_1 l_2 sin(q2)\dot{q2} \end{bmatrix},$$

$$\mathrm{G} = \begin{bmatrix} m_1 l_1 g cos(q1) + m_2 g(l_2 cos(q_1 + q_2) + l_1 cos(q_1)) \\ m_2 g l_2 cos(q_1 + q_2) \end{bmatrix}$$

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

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

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

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

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

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

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

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

We assume that the system has initial positions:

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

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

$$f = K_P e + K_D \dot{e} + K_1 \int edt.$$

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:

$$f_1 = K_{p_1} e_1(q_1) + K_{D_1} \dot{e_1}(q_1) + K_{I_1} \int e_1(q_1) dt = K_{p_1}(q_{1f} - q_1) - K_{D_1} \dot{q}_1 + K_{I_1} \int (q_{1f} - q_1) dt$$
$$f_2 = K_{p_2} e_2(q_1 2 + K_{D_2} \dot{e_2}(q_2) + K_{I_2} \int e_2(q_2) dt = K_{p_2}(q_{2f} - q_2) - K_{D_2} \dot{q}_2 + K_{I_2} \int (q_{2f} - q_2) dt$$

where q_{1f} and q_{2f} are given constants.

The complete system of equations with control is then

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

where

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

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

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

Differentiating with respect to t gives

$\dot{x_1} = e(q_1) = q_{1f} - q_1$, $\dot{x_2} = e(q_2) = q_{2f} - q_2,$

The complete equations are

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

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

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = -M^{-1}(q)[C(q, \dot{q})\dot{q} + G(q)] + \begin{bmatrix} K_{p_1}(q_{1f} - q_1) - K_{d_1}\dot{q}_1 + K_{I_1}x_1 \\ K_{p_2}(q_{2f} - q_2) - K_{d_2}\dot{q}_2 + K_{I_2}x_2 \end{bmatrix}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} K_{p_1}(q_{1f} - q_1) - K_{d_1}\dot{q}_1 + K_{I_1}x_1 \\ K_{p_2}(q_{2f} - q_2) - K_{d_2}\dot{q}_2 + K_{I_2}x_2 \end{bmatrix}$$

To discretize the above system of differential equations in time, we transform them into a system of first-order ordinary differential equations. To do this, we define six new variables as follows:

$u_1 = q_1, u_2 = q_2, u_3 = \dot{q_1}, u_4 = \dot{q_2}, u_5 = x_1, u_6 = x_2,$

After differentiating, we have

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

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

$\dot{u_3} = \ddot{q_1} = \phi(t, u_1, u_2, u_3, u_4, u_5, u_6),$

$\dot{u_4} = \ddot{q_2} = \psi(t, u_1, u_2, u_3, u_4, u_5, u_6),$

$\dot{u_5} = \dot{x_1} = q_{1f} - u_1,$

$\dot{u_6} = \dot{x_2} = q_{2f} - u_2$

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

$$\begin{bmatrix} \phi \\ \psi \end{bmatrix} = -M^{-1}(q)[C(q, \dot{(q)})\dot{(q)} + G(q)] + \begin{bmatrix} K_{P_1}(q_{1f} - u1) - K_{D_1}u_3 + K_{I_1}u_5 \\ K_{P_2}(q_{2f} - u2) - K_{D_2}u_4 + K_{I_2}u_6 \end{bmatrix}$$

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

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

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

where $\boldsymbol{U} = [u_1, u_2, u_3, u_4, u_5, u_6]^t$ and $\boldsymbol{H} = [h_1, h_2, h_3, h_4, h_5, h_6]^t$ with

$h_1 = u_3,\ h_2 = u_4,$

$h_3 = \phi(t, u_1, u_2, u_3, u_4, u_5, u_6),\ h_4 = \psi(t, u_1, u_2, u_3, u_4, u_5, u_6),$

$u_5 = q_{1f} - u_1,\ u_6 = q_{2f} - u_2$

The initial conditions are given by:

$\boldsymbol{U_0}$ =[u$_1(0), u_2(0), u_3(0), u_4(0), u_5(0), u_6(0)]^t$,

where,

$u_1(0) = q_1(0), u_2(0) = q_2(0), u_3(0) = \dot{q_1}(0), u_4(0) = \dot{q_2}(0), u_5 = x_1(0), u_6 = x_2(0).$

This system can be solved for the unknown vector U, using the ode45 command defined in MATLAB to solve the system of ordinary differential equations numerically.

Once we solve for, we can obtain torques using $\boldsymbol{U} = [u_1, u_2, u_3, u_4, u_5, u_6]^t$,

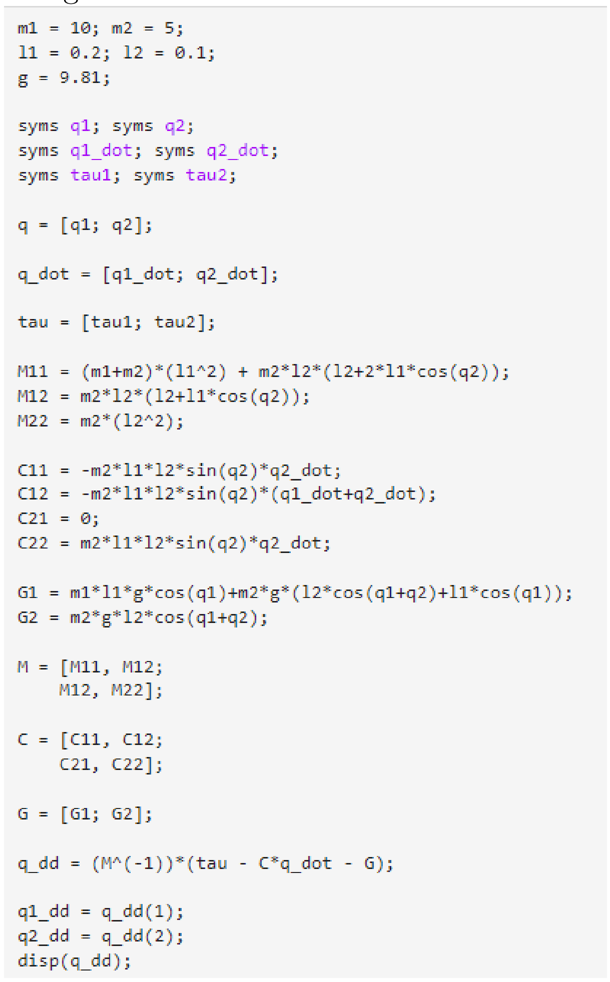
$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = M(q) \begin{bmatrix} K_{P_1}(q_{1f} - u_1) - K_{D_1}u_3 + K_{I_1}u_5 \\ K_{P_2}(q_{2f} - u_2) - K_{D_2}u_4 + K_{I_2}u_6 \end{bmatrix}$$

Chapter 3

METHODOLOGY

3.1 FINDING $\ddot{q}$ MATRIX

The final equations derived above are used to compute the $\ddot{q}$ matrix by writing the following code:-



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

syms q1; syms q2;
syms q1_dot; syms q2_dot;
syms tau1; syms tau2;

q = [q1; q2];

q_dot = [q1_dot; q2_dot];

tau = [tau1; tau2];

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

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;

G1 = m1*l1*g*cos(q1)+m2*g*(l2*cos(q1+q2)+l1*cos(q1));
G2 = m2*g*l2*cos(q1+q2);

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

C = [C11, C12;
    C21, C22];

G = [G1; G2];

q_dd = (M^(-1))*(tau - C*q_dot - G);

q1_dd = q_dd(1);
q2_dd = q_dd(2);
disp(q_dd);
```

The provided code is written in MATLAB and is used to calculate the acceleration of a two-link planar robotic manipulator. The code uses symbolic variables and symbolic

math operations to derive the dynamic equations of motion for the manipulator.
Here's a step-by-step explanation of the above Code:-

3.1.1 PARAMETER DEFINITIONS

The initial lines of code define the masses of the two link manipulators, their respective lengths and acceleration due to gravity as specified in the Problem Statement.

3.1.2 SYMBOLIC VARIABLE DEFINITIONS

syms q1; syms q2;
These lines declare symbolic variables q1 and q2 to represent the joint angles of the manipulator.
syms q_1_dot ; syms q_2_dot;
These lines declare symbolic variables q_1_dot and q_2_dot to represent the joint angular velocities.
syms τ_1; syms τ_2;
These lines declare symbolic variables τ_1 and τ_2 to represent the applied joint torques.

3.1.3 DYNAMIC EQUATION FORMULATION

All the symbolic variable matrices are intialized to their corresponding values.
The Mass Matrix "M", the Coriolis and Centrifugal Matrix "C", and the Gravitational Vector "G" are also set up in accordance with the provided problem description.

3.1.4 FINAL $\ddot{q}$ CALCULATION

Following from the above modelling to solve the dynamic equations of motion, $\ddot{q} = (M^{-1})(\tau - C\dot{q} - G)$ is finally used to calculate the joint accelerations.

3.2 CONTROLLER DESIGN

After computing the $\ddot{q}$ matrix, its value is directly used in the final code:-

```
t_span = [0 10];
q_initial = [0.1; 0.1];
qdot_initial = [0; 0];
x_initial = [0; 0];

q_desired = [0; 0];

[t, states] = ode45(@compute_dynamics, t_span, [q_initial; qdot_initial; x_initial]);
q1 = states(:,1);
q2 = states(:,2);
q1_dot = states(:,3);
q2_dot = states(:,4);
x1 = states(:,5);
x2 = states(:,6);

e1 = q_desired(1) - q1;
e2 = q_desired(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,q2);
title("q_2");
xlabel("t");
ylabel("q_2(t)");

figure; subplot(2,1,1);
plot(t,e1);
title("Error in q_1");
xlabel("t");
ylabel("e_1(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 = 200; Kd1 = 150; Ki1 = 100;
    % Kp2 = 10; Kd2 = 0; Ki2 = 10;
    Kp2 = 200; 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*(l2+l1*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*sin(q2)*(q1_dot + q2_dot))/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.

3.2.1 INITIALIZATION

t_span = [0 10], this sets the time span for the simulation from 0 to 10 seconds. q_initial, qdot_initial, and x_initial define the initial joint angles, joint velocities, and control variables for the simulation.

x denotes the integral of error, which is the difference in the final equilibrium state and the current angle.

$$x_1 = \int e_1 \, dt, \tag{3.1}$$

where e_1 denotes the error in angle q_1 and

$$x_2 = \int e_2 \, dt, \tag{3.2}$$

where e_2 denotes the error in angle q_2

q_desired represents the desired joint angles that the controller aims to achieve. The final q_desired value should be 0 for both q_1 and q_2.

3.2.2 ODE INTEGRATION

ode45 is a MATLAB function for solving ordinary differential equations (ODEs) using the specified function compute_dynamics.It integrates the dynamics of the manipulator over time, starting from the initial conditions provided.The results are stored in t(time) and $states$(joint angles, joint velocities, and control variables).

3.2.3 PID CONTROL FUNCTION COMPUTE_DYNAMICS

This function is defined to compute the dynamics of the two-link manipulator and apply PID control. It takes the current time t and the state vector states as input arguments.

3.2.4 PID CONTROL GAINS

Kp1, Kd1, and Ki1 represent the proportional, derivative, and integral gains for the first joint (q_1).

Kp2, Kd2, and Ki2 represent the gains for the second joint (q_2).
These gains are used to calculate the control forces applied to each joint.

The control forces f_1 and f_2 for each joint are calculated using PID control with proportional, derivative, and integral terms. These control forces aim to minimize the joint angle errors. The control forces are used to compute the applied torques on the joints (tau1 and tau2).

3.2.5 STATE VARIABLES

The function returns a vector dxdt that represents the time derivatives of the states. These derivatives are used by ode45 to integrate the dynamics over time.

- dxdt(1) represents the derivative of q1
- dxdt(2) represents the derivative of q2
- dxdt(3) is the double derivative of q1
- dxdt(4) is the double derivative of q2
- dxdt(5) denotes the error in q1
- dxdt(6) denotes the error in q2

These derivatives are instrumental in describing the system's dynamic behavior and are essential for the integration of the robotic manipulator's dynamics over time using ode45.

3.2.6 PLOTS

The code then generates plots of joint angles (q_1 and q_2) and errors (e_1 and e_2) over time using MATLAB's subplot and plot functions.

In summary, this code sets up and simulates the control of a two-link planar manipulator using PID control. It computes the dynamics of the manipulator, calculates control forces, and evaluates the performance of the controller by plotting joint angles and errors over time.

Chapter 4

OBSERVATIONS AND RESULTS

The two-link manipulator's controller is tuned to optimize the values of Kp (proportional gain), Kd (derivative gain) and Ki (integral gain) to achieve the desired performance in terms of stability, steady state error, transient response and responsiveness of the system. The control system should be stable and have less error. It should not exhibit sustained oscillations or diverge uncontrollably.

Transient response characteristics encompass rising time, settling time, and overshoot. Reducing overshoot enhances precision but may introduce sluggishness if Kp is significantly reduced. Less overshoot indicates that the system approaches the desired setpoint smoothly and doesn't exceed it by a significant margin. Increasing overshoot results in a faster response. The steady error of the system should be minimized if the system needs to maintain a specific setpoint accurately.

Role of Kp Excessively high Kp values can lead to instability, causing oscillations or even system divergence. P-only control minimizes the fluctuation in the process variable, but it does not always bring the system to the desired set point. It provides a faster response than most other controllers, initially allowing the P-only controller to respond a few seconds faster. However, as the system becomes more complex (i.e. more complex algorithm) the response time difference could accumulate, allowing the P-controller to possibly respond even a few minutes faster. Although the P-only controller does offer the advantage of faster response time, it produces deviation from the set point. This deviation is known as the offset, and it is usually not desired in a process. Reducing Kp makes the control action less aggressive, resulting in less overshoot. However, if Kp is reduced highly, it may lead to slow or sluggish responses.

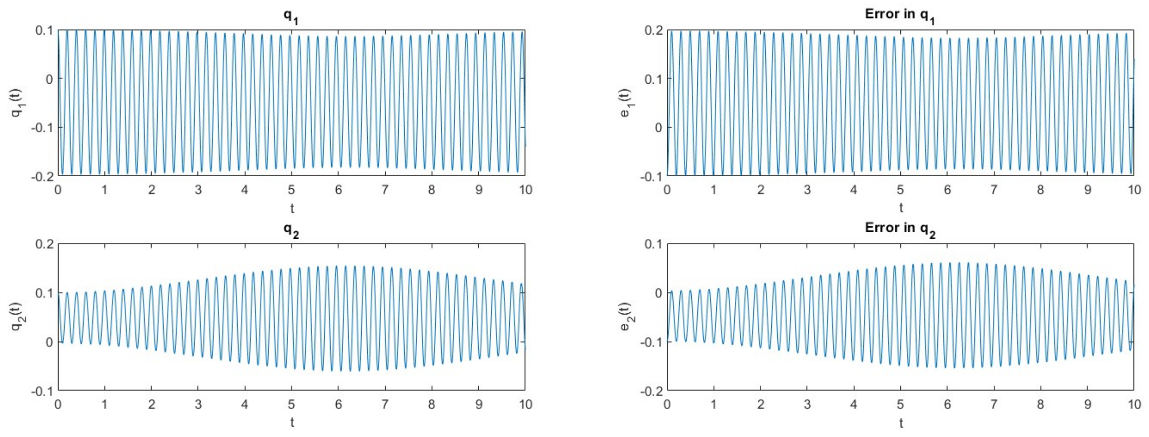
Role of Ki Integral control (Ki) is responsible for eliminating steady-state error. Increasing Ki leads to faster error correction, which minimizes steady-state deviations.The key advantage of adding a I-control to your controller is that it will eliminate the offset. The disadvantages are that it can destabilize the controller,however,I-only controllers are much slower in their response time.But it can contribute to overshoot if set too high. Incremental adjustments to Ki should be made to ensure that it does not lead to excessive overshoot.

Role of Kd Derivative control (Kd) helps dampen oscillations and reduce oscillations by predicting how fast the error is changing. Increasing Kd can help stabilize the system and mitigate overshoot. Kd can help reduce settling time by reducing oscillations. It allows the system to reach the setpoint more quickly without excessive oscillatory behaviour.

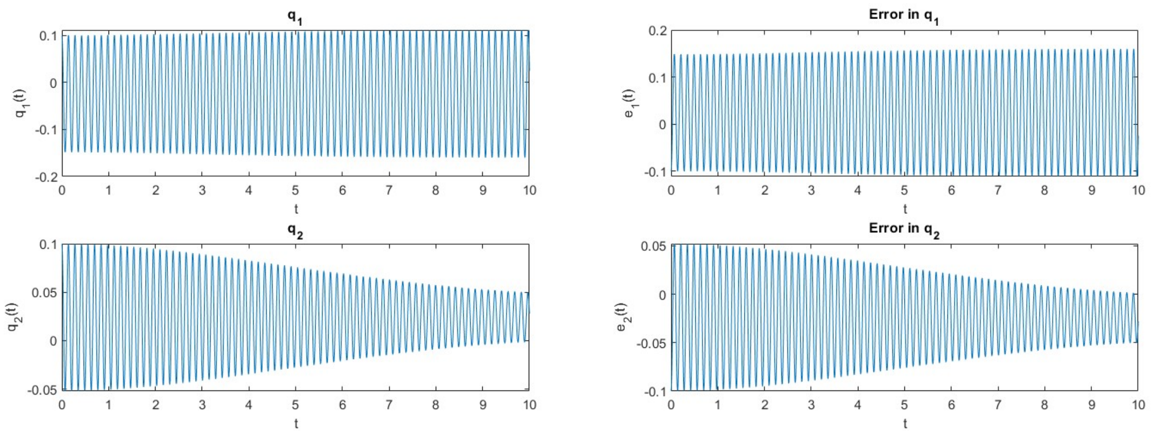
4.1 TUNING THE SYSTEM MODEL TO OPTIMIZE KP,KD AND KI

We begin by setting both Ki and Kd to zero. Kp is initially set to a small value and then gradually increased. Increasing Kp makes the control action more aggressive. If Kp is highly increased, the system becomes unstable and exhibits excessive oscillations. Reducing Kp adds more damping to the system response. The system response becomes slower as Kp is reduced. We try to strike a balance between overshoot and settling time by considering a compromise value for Kp.

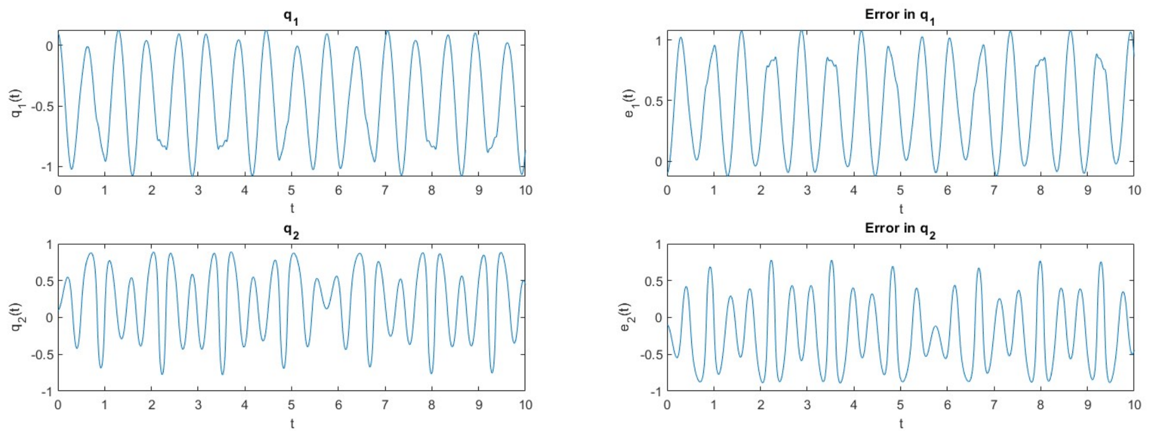
For Kp = 1000,



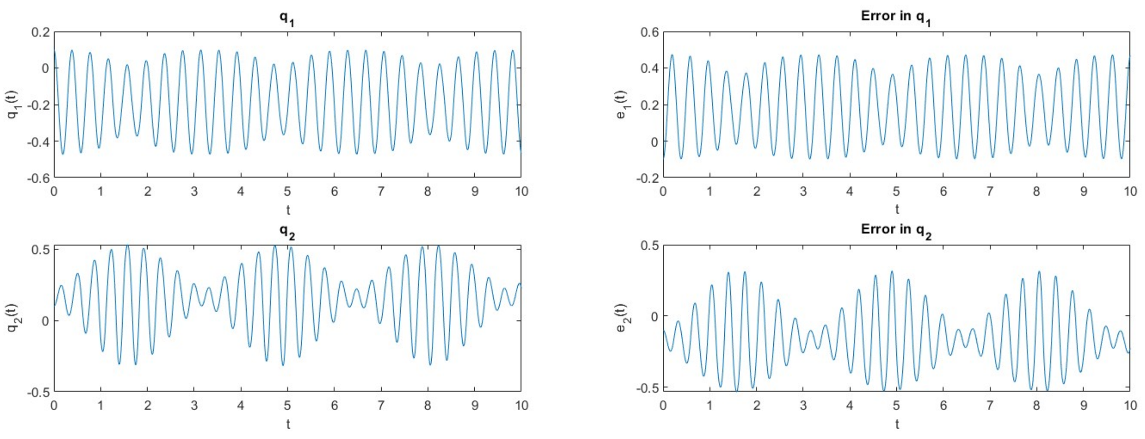
For Kp = 2000,



For Kp = 80,



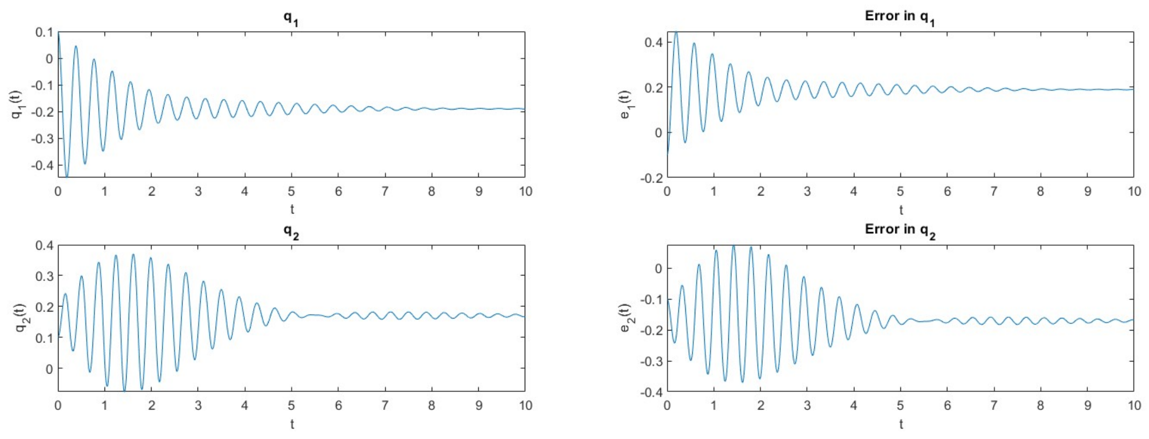
For Kp = 250,



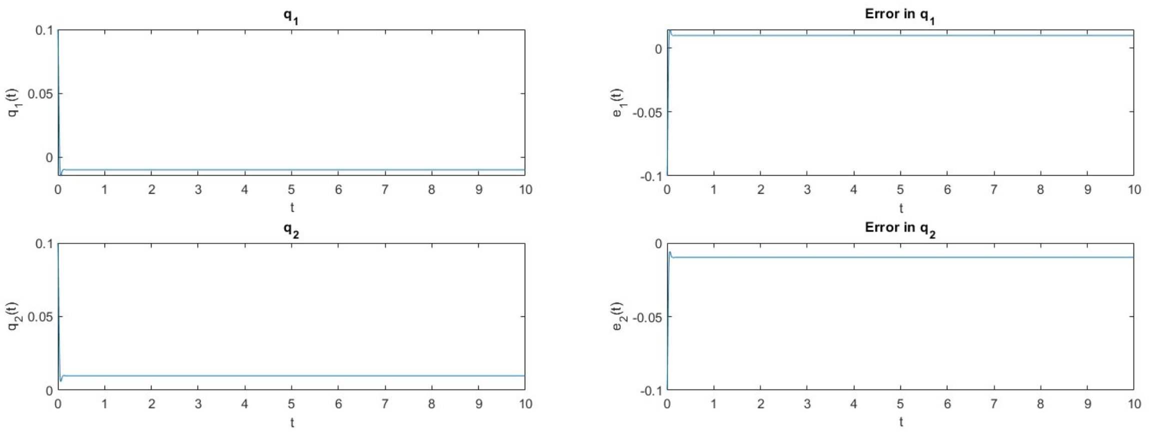
4.1.1 PD CONTROLLER

We assign a small value to Kd initially while keeping Ki zero and a suitable value for Kp. We observe the system's response to reference setpoint disturbances while paying attention to overshoot and oscillations in the response. As Kd is increased, we see a reduction in oscillations and overshoot. Kd helps add damping to the system, which can improve the transient response and reduce oscillatory behaviour. Excessively high Kd value can lead to instability and noise amplification in the control system. We continue to adjust Kd until we find a balance between reducing overshoot and maintaining stability. The optimal Kd value result in a controlled and well-damped response without excessive oscillations. We have finally set our Kp value to 5000 and Kd value to 100 for PD controller.

For Kp = 250, Kd = 1,



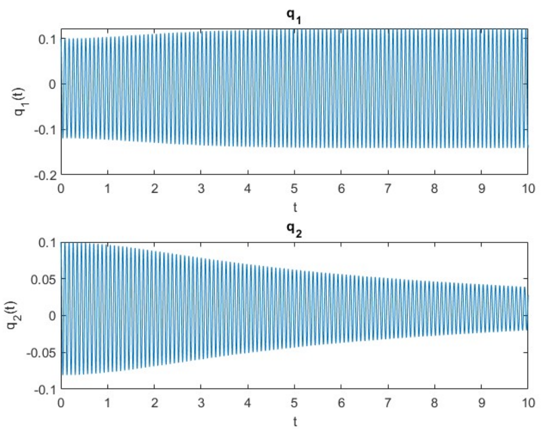
For Kp = 5000, Kd = 100,

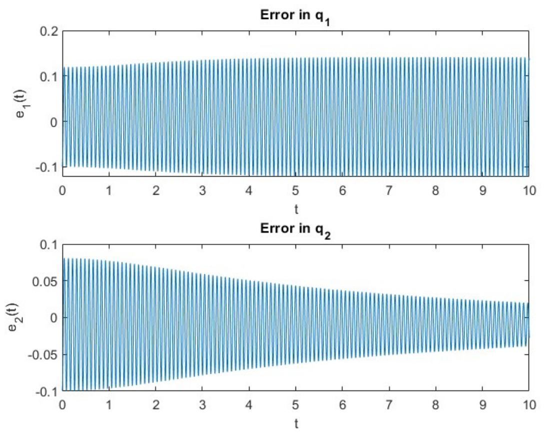


4.1.2 PI CONTROLLER

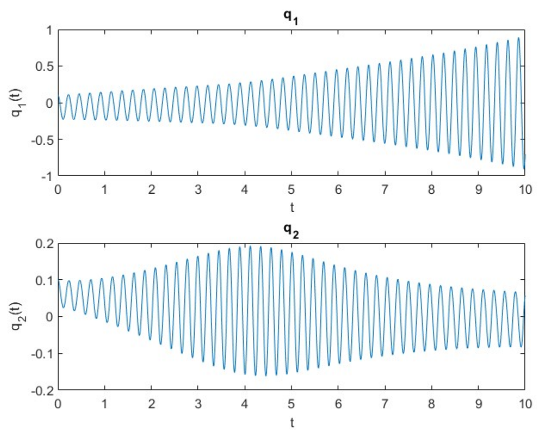
We assign a small value to Ki initially while keeping Kd zero and a suitable value for Kp. We observe the system's response to reference setpoint disturbances while paying attention to the steady state error. On increasing Ki, the steady state error starts reducing and the system starts tracking the setpoint more accurately. An excessively high value of Ki can lead to excessive control action, which may result in instability or oscillations. We were unable to compute any optimal value of Kp and Ki such that the system response was stable and steady state error was minimized. As Ki was increased, the initial value of error was less but the error was diverging in nature. Two-link manipulators often have nonlinear dynamics, including friction, gravity, and coupling between joint motions. A PI controller may struggle to handle such nonlinearity effectively, leading to oscillations and steady-state errors. We have set our Kp value to 750 and Ki value to 250 for PI controller.

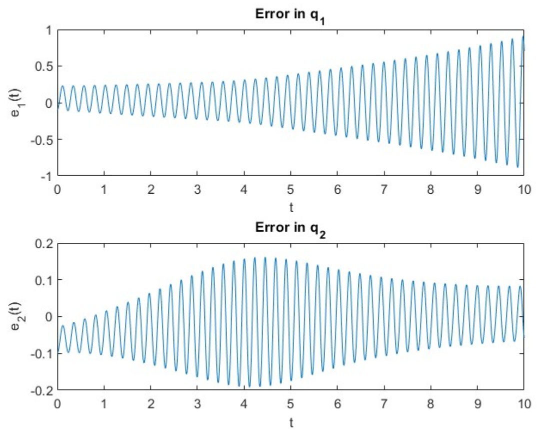
For Kp = 5000, Ki = 10,



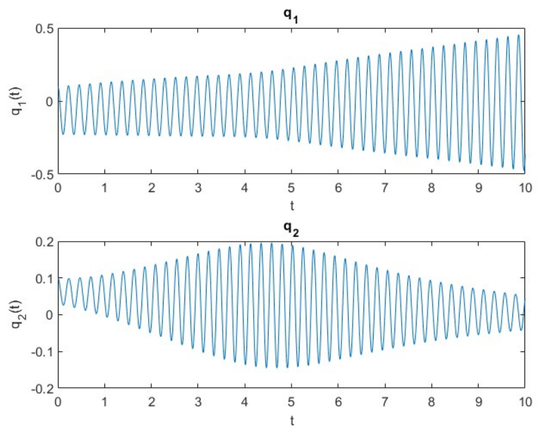


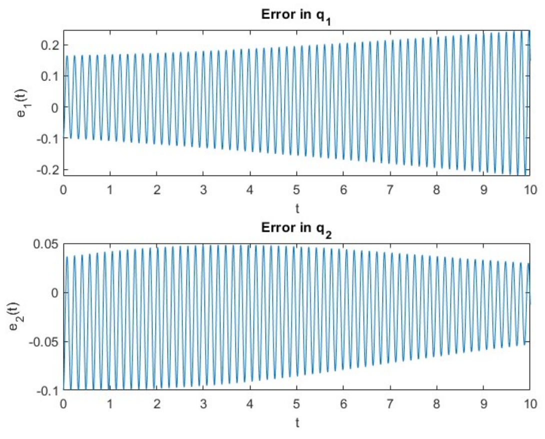
For Kp = 750, Ki = 250,



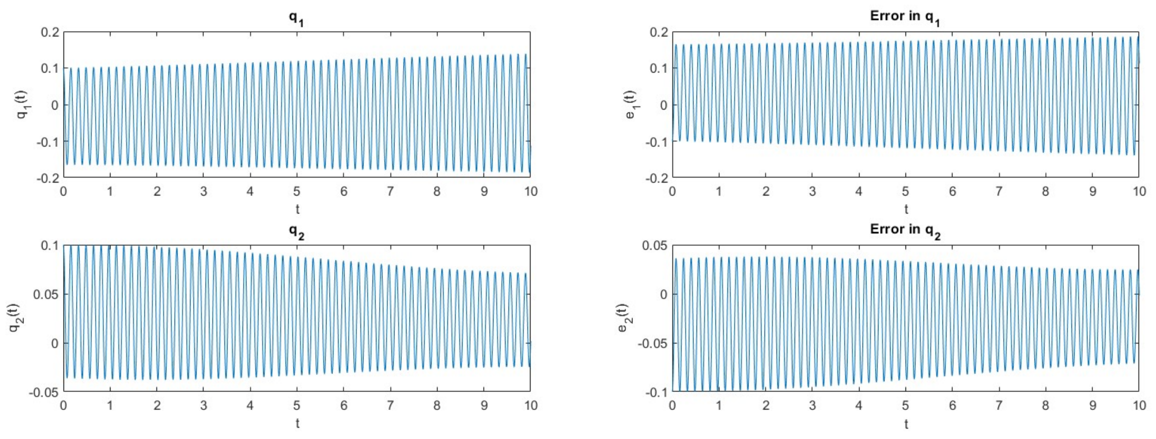


For Kp = 1500, Ki = 150,





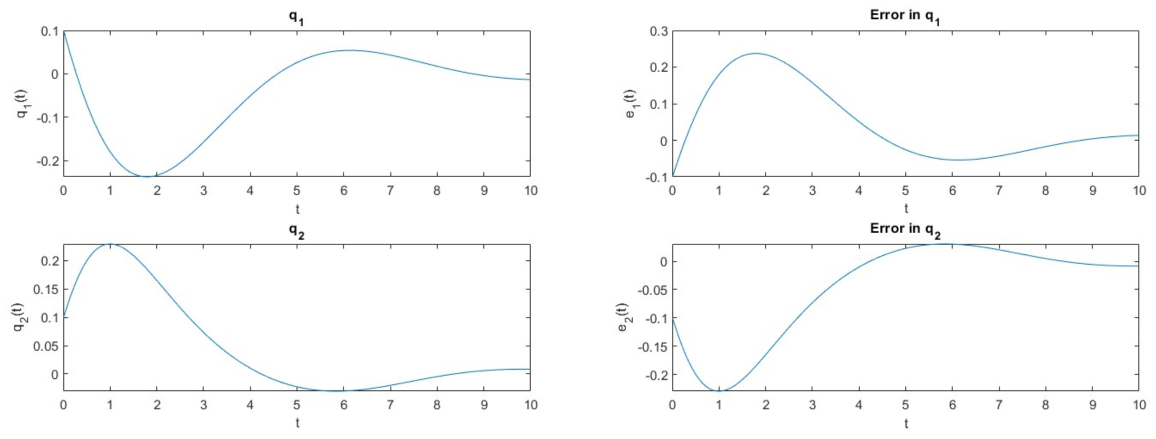
For Kp = 1500, Ki = 50,



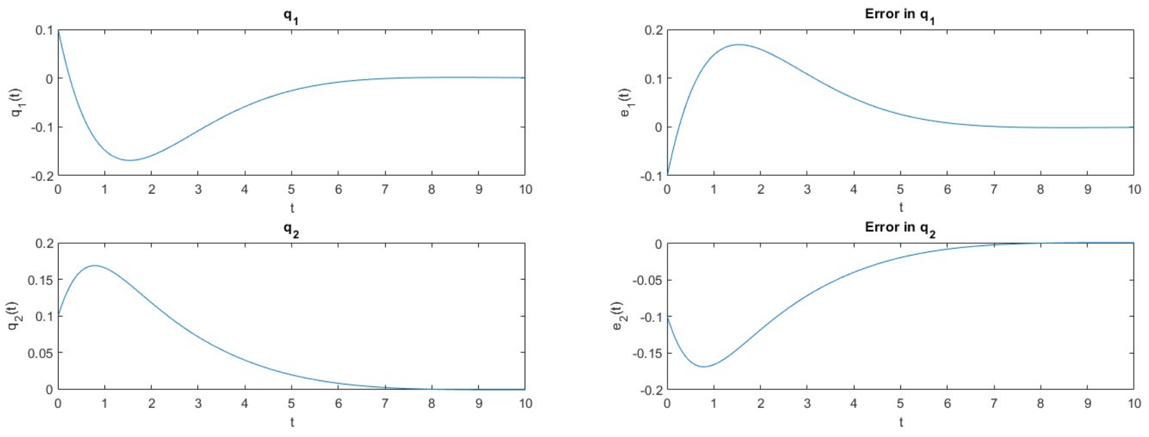
4.1.3 PID CONTROLLER

We assign a suitable value for Kp by keeping a balance between overshoot and settling time and setting both Kd and Ki to zero. We then gradually increase Kd to reduce overshoot and improve damping. We continue to adjust Kd until we find a balance between reducing overshoot and maintaining stability. We obtain a highly stable system response with quite less overshoot and settling time but some steady state error. To reduce this steady state error, we gradually increase Ki until we obtain a system response with negligible steady state error. Thus, we obtain a highly stable system response with negligible error which attains the final state quickly without deviating much from the desired path.We have set out Kp value to 250, Kd value to 150 and Ki value to 100 for PID controller.

For Kp = 100, Kd = 100, Ki = 100,



For Kp = 200, Kd = 150, Ki = 100,



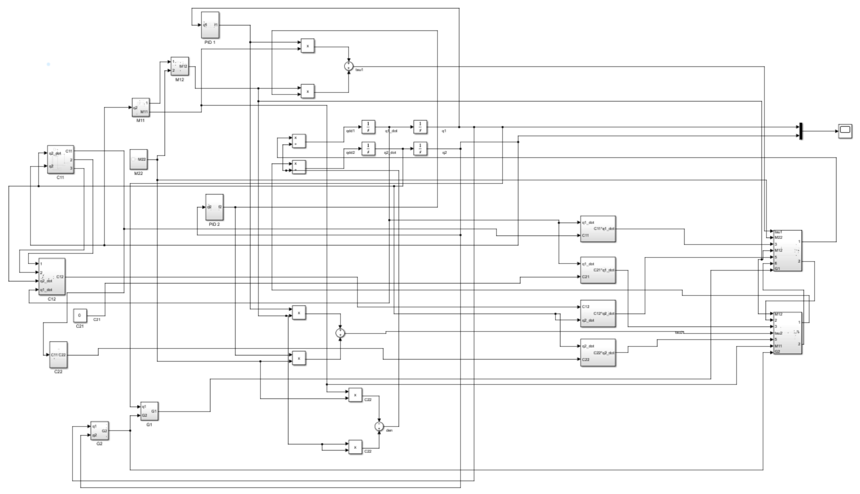
4.1.4 FINAL RESULTS

The PID control method stands out as the most effective approach for modelling and controlling a two-link manipulator. PID controllers combine proportional, integral, and derivative control actions, offering a balanced and versatile solution. While PID provides accurate trajectory tracking and steady-state accuracy, it addresses several issues often encountered with PD and PI control methods. PD control, while effective in reducing overshoot, struggles with eliminating steady-state errors. In contrast, PI control excels at steady-state error correction but often exhibits overshoot and sluggish responses. The PID controller's integration of all three control components allows it to address these challenges comprehensively, making it a preferred choice for modelling and controlling two-link manipulators in diverse applications, offering a balance between accuracy, stability, and response speed.

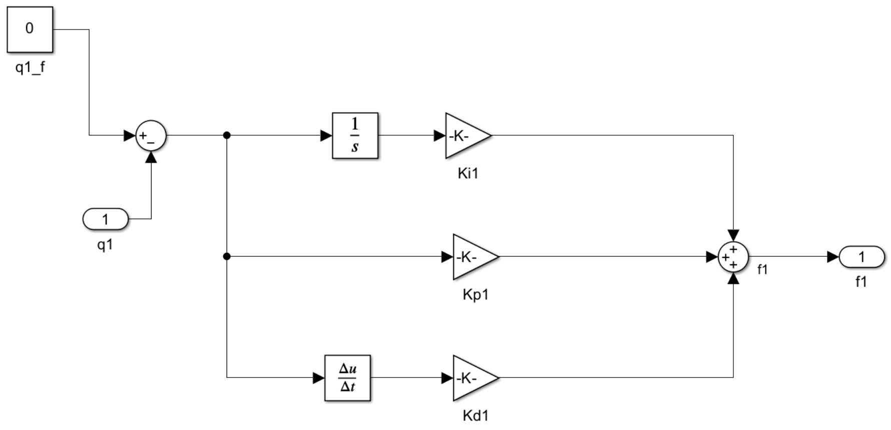
Chapter 5

SIMULINK

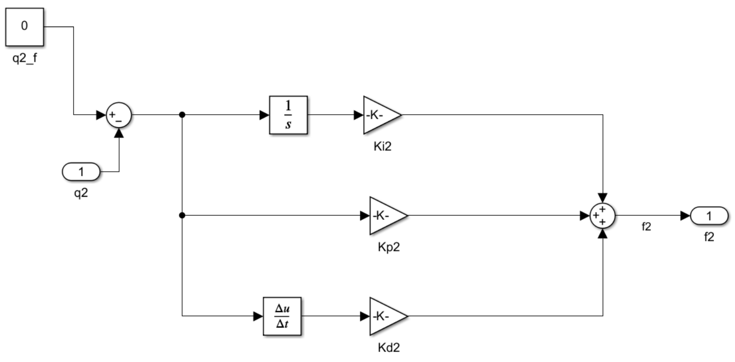
5.1 SYSTEM DESIGNED IN SIMULINK



5.1.1 PID SYSTEM-1

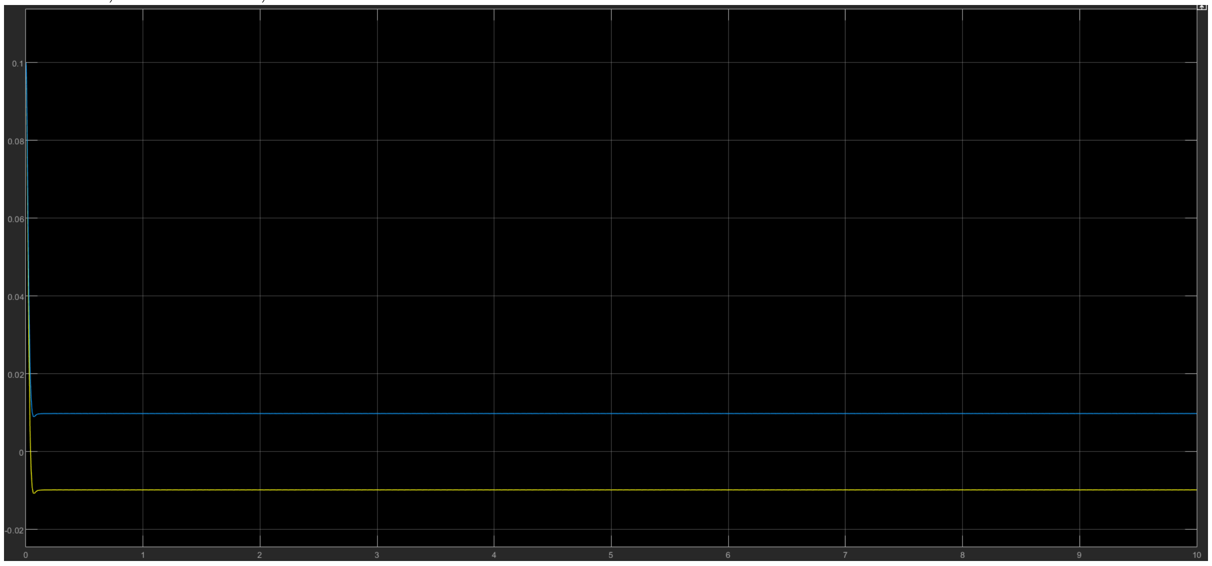


5.1.2 PID SYSTEM-2



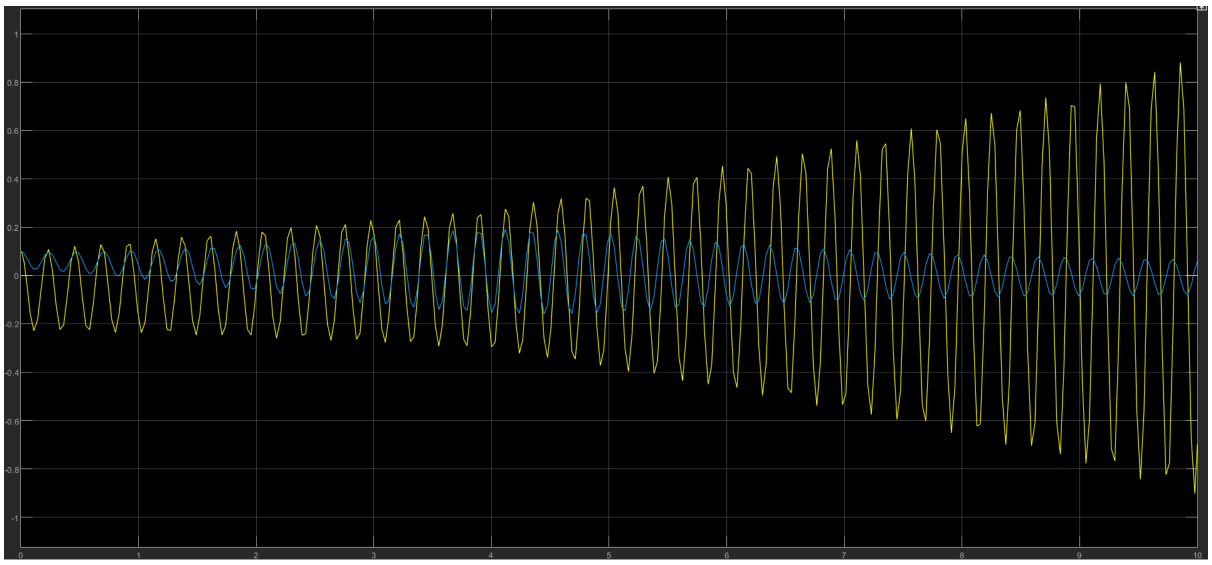
5.2 FINAL PD CONTROL

Kp = 5000, Kd = 100, Ki = 0



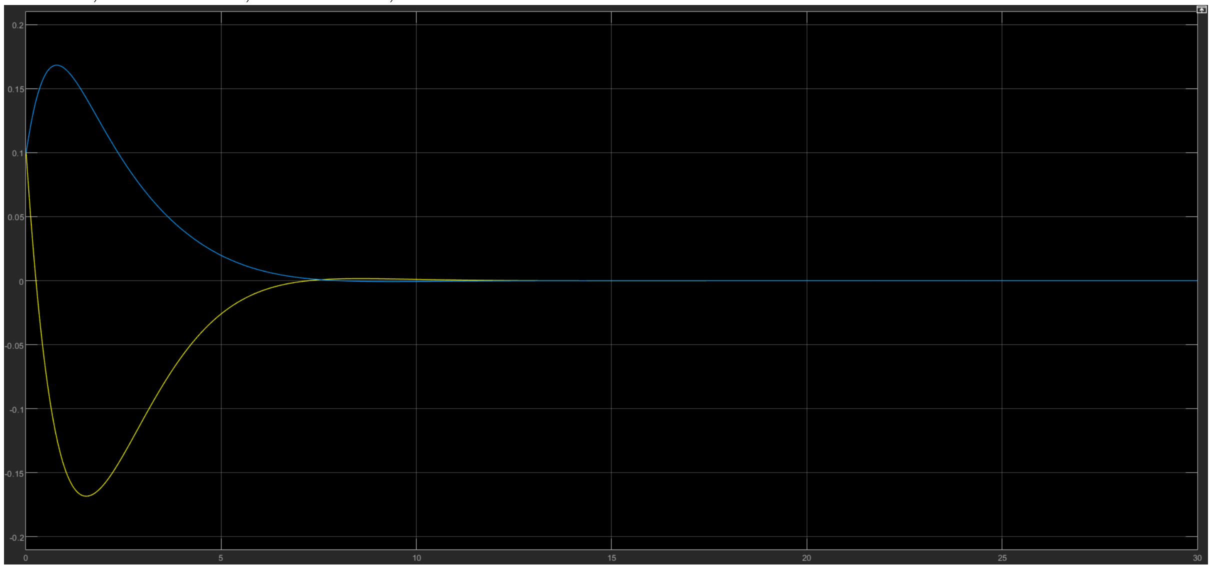
5.3 FINAL PI CONTROL

Kp = 750, Ki = 250, Kd = 0,



5.4 PID CONTROL

Kp = 200, Kd = 150, Ki = 100,



Chapter 6

CONCLUSION

In this report, a two-link robotic manipulator is discussed and its dynamics were modeled using Lagrange mechanics. First, we derived the dynamical equation of the two-link robot manipulators using Euler-Lagrange method and then a simple and efficient control scheme was developed. More specifically,a Proportional controller,Proportional derivative controller and Proportional-Integral-Derivative (PID) controller was introduced to control the motion of the robot at a specific position. In particular, we described how PD, PI and PID controller can be used to keep the links in a desired position. The efficiency of PD, PI and PID controller is verified by the simulation results. This model can be used to design and code control algorithms for a mobile robot.In this project, we considered a planar robot having rotational joints. The performance of two-link robotic manipulator is investigated with PD, PI and PID control. The PID controller resulted in the best performance and very effective and accurate trajectory tracking capability as compared to PD controller. Also the response with PID controller was having reduced oscillations about the desired trajectory as compared to PD and PI controller.