Jaypee Institute of Information Technology, Sector - 62, Noida

Mathematics Project Report

Project Report

Detailed Report on Utilizing Laplace transforms for Stability Analysis in Feedback Control Systems for Robotics

Submitted to

Dr. Pato Kumari
Mathematics Department

Submitted by

Aarnya Jain 2401030209
Vansh Garg 2401030202
Amolik Agarwal 2401030193
Paras Mehta 2401030210
Yatin Khatri 2401030214

Letter of Transmittal

Dr. Pato Kumari

Department of Mathematics

Jaypee Institute of Information Technology

Subject: Utilizing Laplace transforms for Stability Analysis in Feedback Control Systems for Robotics.

Dear Ma'am,

I am pleased to submit my report on Utilizing Laplace transforms for Stability Analysis in Feedback Control Systems for Robotics of AI at Jaypee Institute of Information Technology. This report aims to provide an in-depth analysis of Laplace Transformations in Feedback Control Systems. Thank you for trusting us to complete this report for you.

Sincerely,

Aarnya Jain 2401030209
Vansh Garg 2401030202
Amolik Agarwal 2401030193
Paras Mehta 2401030210
Yatin Khatri 2401030214

Date: November 30, 2024

Contents

1 Summary

This report explores the application of Laplace Transforms in the analysis and design of feedback control systems. Laplace Transforms are a powerful mathematical tool that facilitates the transition from time-domain to s-domain analysis, simplifying the study of linear systems. The report outlines the fundamental concepts of Laplace Transforms, including the transform of common functions, the inverse transform, and its application in solving differential equations.

2 Introduction

Feedback control systems are integral to ensuring that the performance of dynamic systems meets specific desired objectives, such as stability, accuracy, and robustness. These systems are designed to automatically adjust the behaviour of a process based on the comparison between the desired output and the actual output. Feedback control is widely used in various fields, including **engineering**, **robotics**, and **artificial intelligence (AI)**.

Laplace transformation plays a critical role in the analysis and design of control systems by providing a mathematical framework that **simplifies the handling of differential equations** and offers insights into **system stability, performance, and time-domain behaviour.** In the context of AI systems, Laplace transforms can help optimize feedback loops, improve decision-making algorithms, and enhance real-time control.

This report explores the use of Laplace transformations in feedback control systems, focusing on their application in AI-based control systems.

3 The Role of Feedback Control Systems in AI

Feedback control systems are designed to ensure that an AI-controlled system performs a task efficiently by continuously monitoring and adjusting its outputs. These systems typically consist of:

- **Plant (or Process):** The dynamic system that is being controlled (e.g., a robotic arm, autonomous vehicle, or a drone).
- **Controller:** The component responsible for adjusting the input to the system to achieve the desired output (e.g., Proportional-Integral-Derivative (PID) controller).
- **Feedback Loop:** A mechanism for comparing the output of the system to the desired goal and making adjustments accordingly.

AI enhances control systems by providing the capability to learn and adapt to new situations. For example, machine learning algorithms can be used to tune the parameters of traditional controllers like PID or develop entirely new control strategies based on data. However, regardless of the complexity of the AI algorithm, the core functionality of feedback control remains rooted in the mathematical analysis of system dynamics, where Laplace transforms are used extensively.

4 Laplace Transformation

Laplace transformation is a powerful mathematical tool used to convert a function of time, typically expressed as a differential equation, into an algebraic equation in the Laplace domain. This conversion simplifies the analysis of dynamic systems, especially in control systems, as it allows for the representation of system behaviour in terms of transfer functions.

The Laplace transforms of a function $f(t)$ is defined as:

$$\int_0^\infty e^{-st} f(t) dt = L\{f(t)\} \tag{1}$$

Let:

$$F(s) = \int_0^\infty e^{-st} f(t) dt = L\{f(t)\} \tag{2}$$

where:

- s is a complex variable, $s = \sigma + i\omega$
- σ represents the real part (decay rate)
- ω represents the imaginary part (frequency)
- $f(t)$ is the time-domain function
- $F(s)$ is the transformed function in the s-domain

In the context of feedback control, the Laplace transform is used to model the behavior of systems and to derive the system's transfer function, which relates the input to the output in the s-domain.

5 Application of Laplace Transformation in Feedback Control Systems

5.1 Transfer Functions and System Modelling

A key concept in feedback control systems is the **transfer function**, which defines the relationship between the system's input and output.

The transfer function $G(s)$ is derived from the Laplace transform of the system's differential equations.

This function is typically expressed as a ratio of polynomials in s:

$$G(s) = \frac{Y(s)}{U(s)} \tag{3}$$

where:

- $Y(s)$ is the Laplace transform of the output function $y(t)$
- $U(s)$ is the Laplace transform of the input function $u(t)$

5.2 Stability Analysis

5.2.1 Poles of a Transfer Function

The **poles of a transfer function** are the values of the complex variable s at which the denominator of the transfer function becomes zero. These poles play a crucial role in determining the behavior and stability of a dynamic system.

Physical Meaning of Poles

1. **System Response**: The poles determine how the system responds to inputs over time. They directly relate to characteristics like oscillations, decay rates, and stability.

2. **Exponential Terms**: In the time domain, the poles define the exponential terms of the system's response (e.g., $e^{\Re(s)t}$ for the real part of a pole).

[Previous content remains the same, continuing from where we left off]

5.2.2 Types of Poles and their Effects

Real Poles

- Negative real poles ($s = -a$) result in an exponentially decaying response, contributing to system stability.

- Positive real poles ($s = +a$) lead to an exponentially growing response, causing instability.

Complex Poles

- Poles with real and imaginary parts ($s = \sigma \pm i\omega$) result in oscillatory behavior.

- The real part (σ) determines the rate of growth/decay of the oscillation.

- The imaginary part (ω) defines the frequency of the oscillation.

For a system to be stable:

- All poles must lie in the **left-half plane** of the complex s-plane, meaning their real parts must be negative ($\Re(s) < 0$).

- If any pole has a positive real part ($\Re(s) > 0$), the system becomes unstable, as responses grow over time.

- If poles are purely imaginary ($\Re(s) = 0$), the system is marginally stable and oscillatory.

6 Sample Problem: Stability Analysis of a Robotic Arm

6.1 Problem Description

The robotic arm must maintain a stable position after being displaced. The control system applies a restoring torque based on the arm's position and velocity.

6.2 Mathematical Modeling

6.2.1 Dynamics of the System

Define the system's differential equation using Newton's Laws:

$$J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} + k\theta = 0 \tag{4}$$

where:

- J: Moment of inertia of the arm
- b: Damping coefficient
- k: Stiffness coefficient
- $\theta(t)$: Angular displacement

6.2.2 Adding a PD Controller

The PD controller modifies the torque:

$$T_c = K_d\frac{d\theta}{dt} + K_p\theta \tag{5}$$

where:

- K_p: Proportional gain
- K_d: Derivative gain

Substituting T_c into the system gives:

$$J\frac{d^2\theta}{dt^2} + (b + K_d)\frac{d\theta}{dt} + (k + K_p)\theta = T_c(t) \tag{6}$$

6.3 Application of Laplace Transform

6.3.1 Transform the Differential Equation

Taking the Laplace transform of the above equation with initial conditions:

$$\left.\frac{d\theta}{dt}\right|_{t=0} = 0 \quad \& \quad \theta(0) = 0 \tag{7}$$

Using Laplace Transformation, we get our input torque equation:

$$\Theta(s)\left[Js^2 + (b+K_d)s + (k+K_p)\right] = T_c(s) = \text{Input}(s) \tag{8}$$

where $\Theta(s)$ is the Output Function and the Laplace transformation of $\theta(t)$.
The transfer function is:

$$G(s) = \frac{\Theta(s)}{\text{Input}(s)} = \frac{1}{Js^2 + (b+K_d)s + (k+K_p)} \tag{9}$$

6.4 Stability Analysis

6.4.1 Identify Poles

Poles of $G(s)$ are the roots of the denominator:

$$Js^2 + (b+K_d)s + (k+K_p) \tag{10}$$

Using the quadratic formula:

$$s = \frac{-(b+K_d) \pm \sqrt{(b+K_d)^2 - 4J(k+K_p)}}{2J} \tag{11}$$

Stability Conditions:
For stability, the real parts of all poles must be negative. The nature of the system depends on the discriminant:

$$\Delta = (b+K_d)^2 - 4J(k+K_p) \tag{12}$$

- $\Delta > 0$: Over damped
  - Discriminant: Positive ($\Delta > 0$)
  - Poles: Real and distinct ($s = -a, s = -b$)
  - Damping State: **Over damped**
  - The system does not oscillate and slowly returns to equilibrium
  - Stability: **Stable**, provided $b + K_d > 0$
- $\Delta = 0$: Critically damped
  - Discriminant: Zero ($\Delta = 0$)
  - Poles: Real and repeated ($s = -(b+K_d)/2J$)
  - Damping State: **Critically Damped**
  - The system does not oscillate and returns to equilibrium as quickly as possible
  - Stability: **Stable**, provided $b + K_d > 0$
- $\Delta < 0$: Under damped
  - Discriminant: Negative ($\Delta < 0$)
  - Poles: Complex conjugates ($s = \sigma \pm i\omega$)
  - Damping State: **Under damped**

- The system oscillates, but oscillations decay over time
- Stability: **Stable**, provided $b + K_d > 0$

Thus, the change in K_p and K_d determine the state of dampness and stability of the arm.

7 Simulation and Graph

7.1 This can also be practically understood through a simple simulation

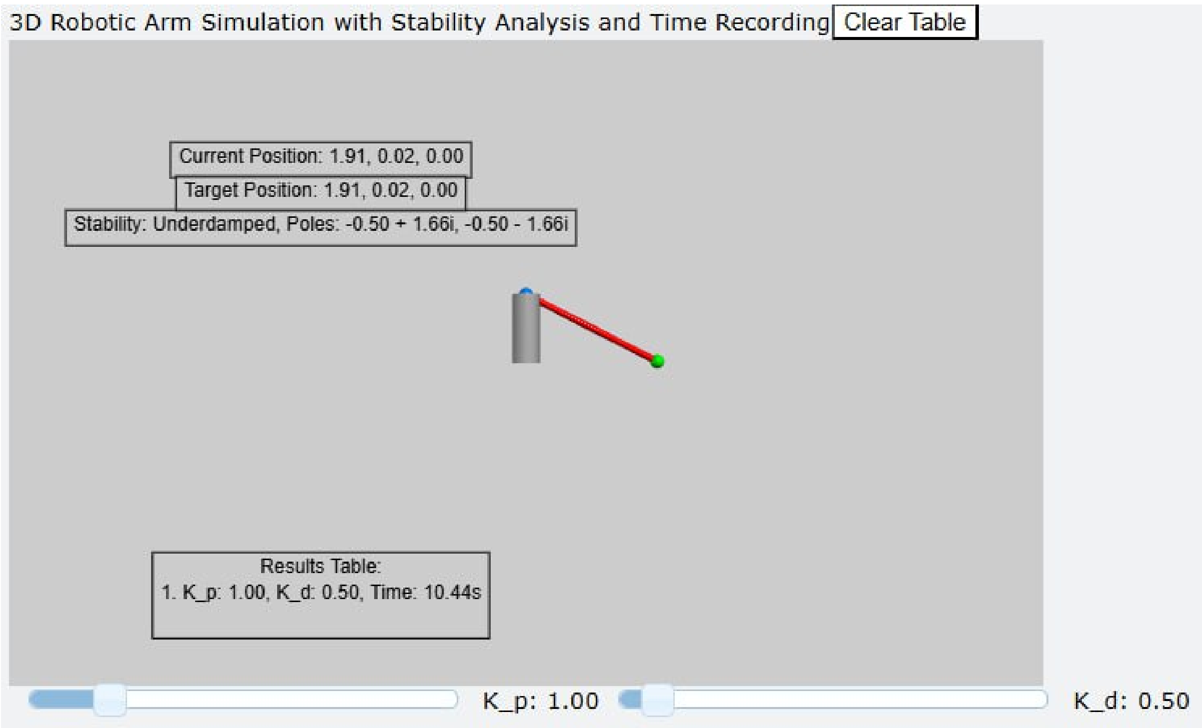

Figure 1: The poles are imaginary thus the system is Under Damped

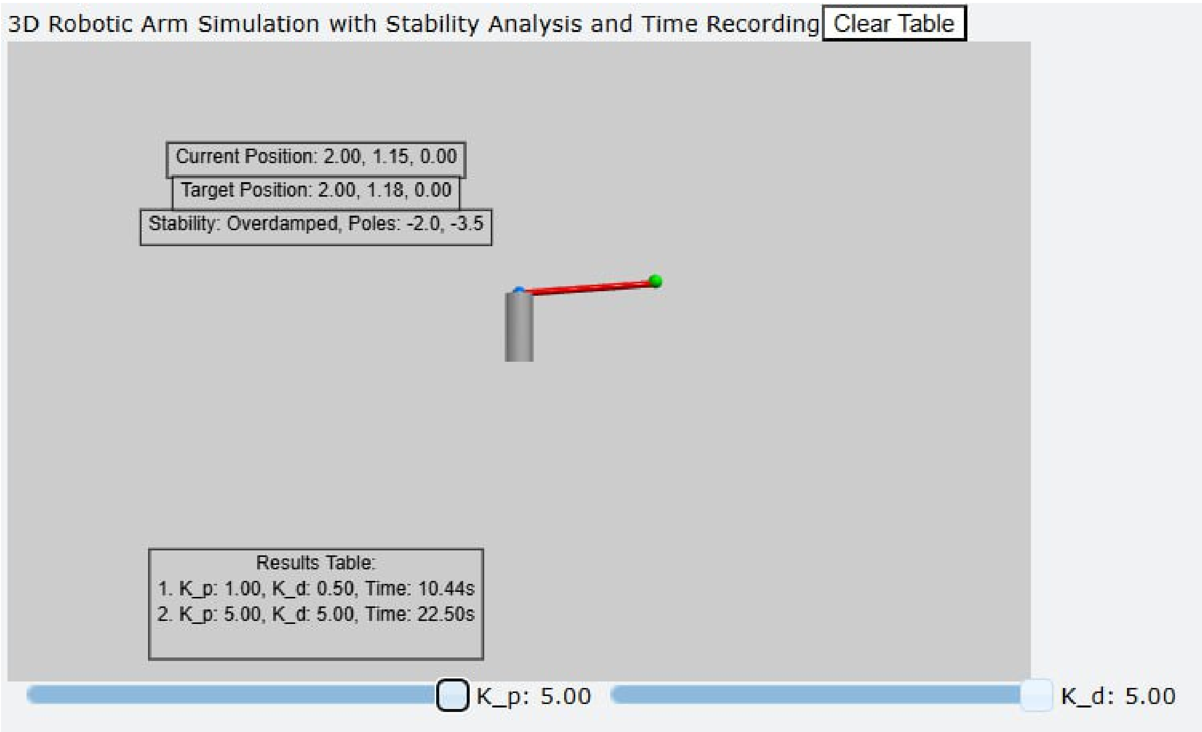

Figure 2: The Poles are real thus the system is Over Damped

The real part is negative in both the cases so the system is stable.

7.2 Analysing through graphs

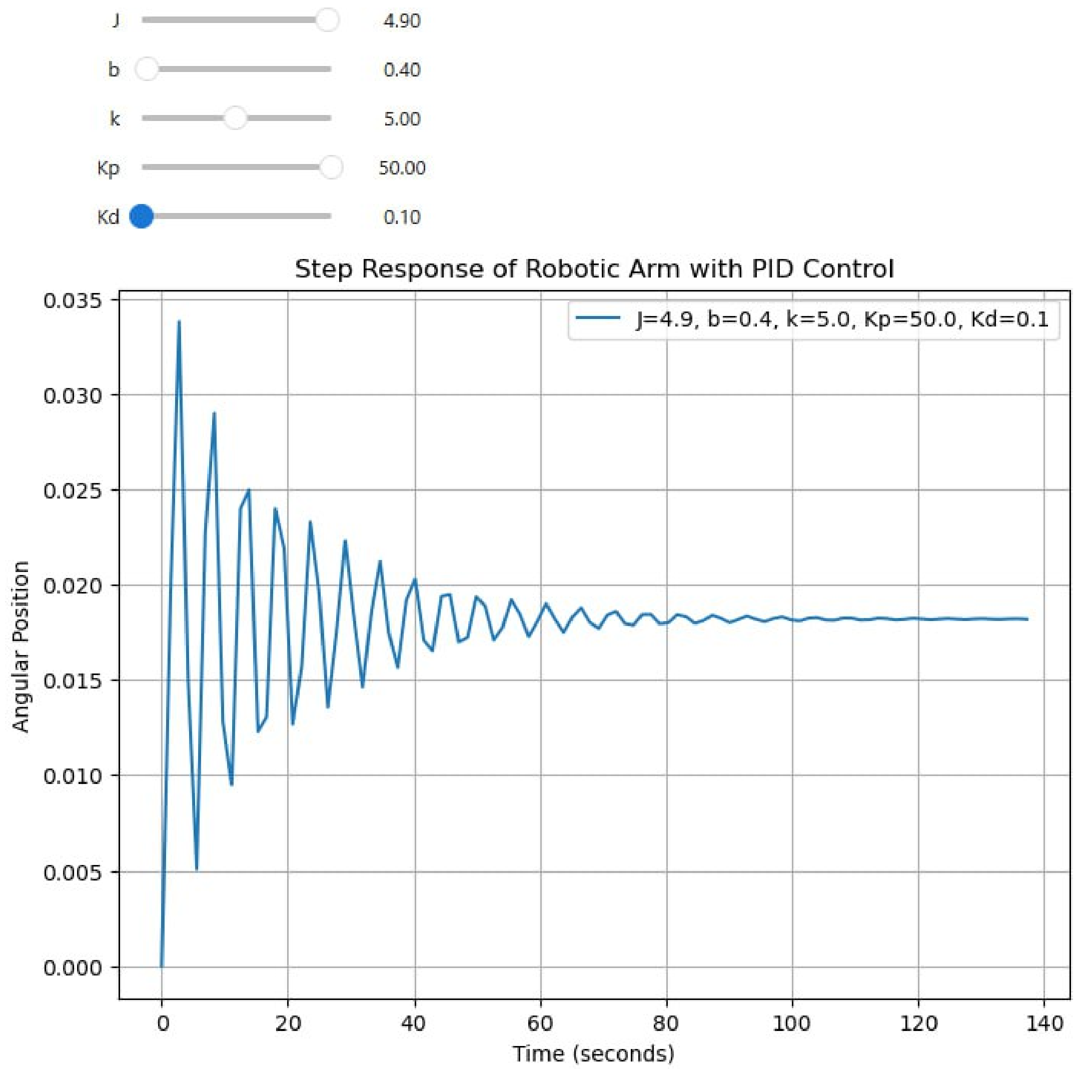

Figure 3: Highly unstable system - $Kp >>> Kd$

7.3 Analysing through graphs

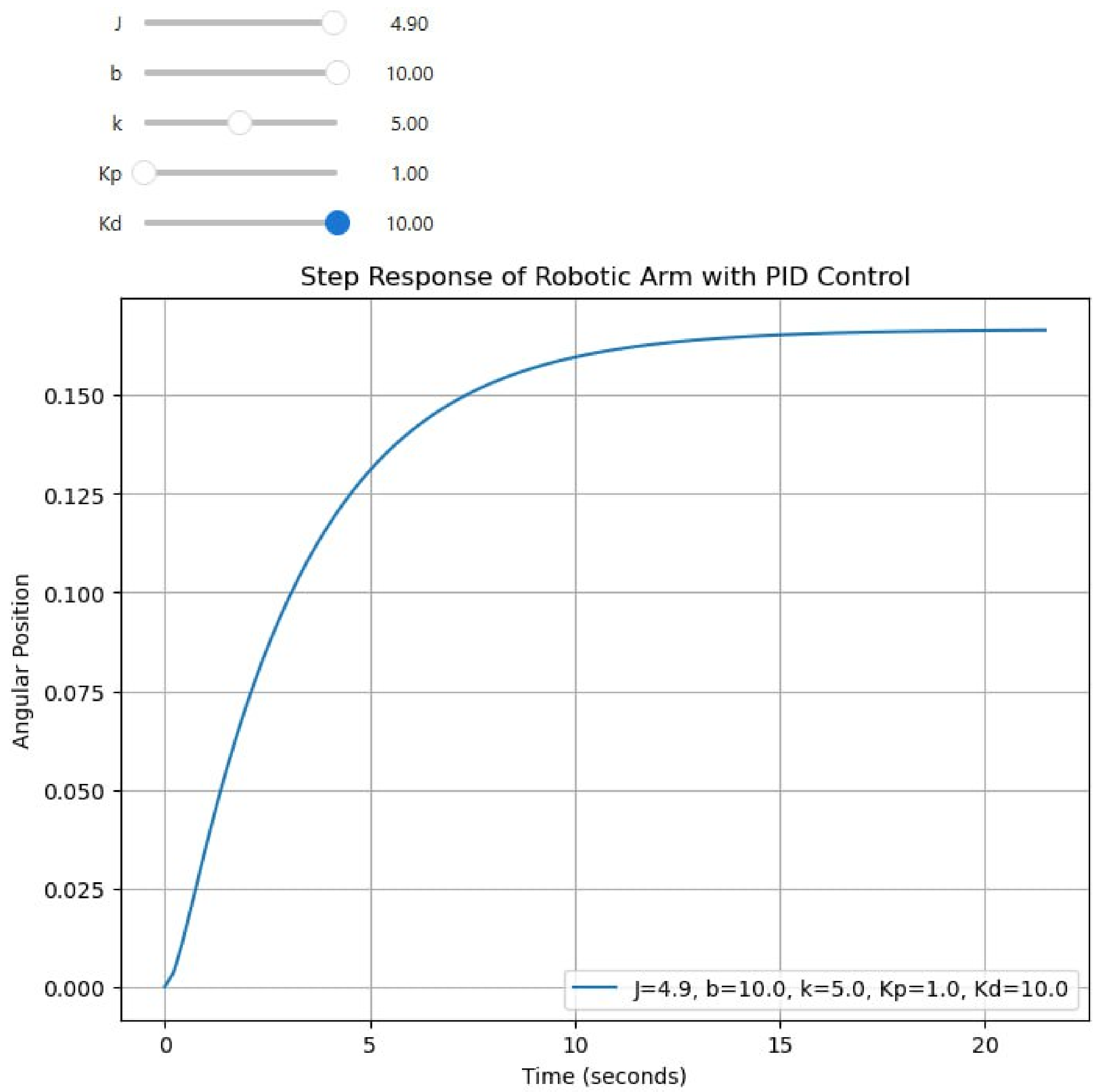

Figure 4: Highly stable system - $Kp <<< Kd$

Thus on analysing the graphs we get that

- Stability is increasing on increasing the damping coefficient b.
- System is highly unstable when $Kp >>> Kd$
- System is highly stable when $Kp <<< Kd$

7.4 Code for the 3D simulation

```python
from vpython import sphere, cylinder, vector, label, rate, scene, slider,
    wtext, button, color
import time
import math

# Set up the 3D scene
scene.title = "3D Robotic Arm Simulation with Stability Analysis and Time
    Recording"
scene.background = vector(0.8, 0.8, 0.8)

# Define Robot Arm Components
base = cylinder(pos=vector(0, 0, 0), axis=vector(0, 1, 0), radius=0.2, color=
    vector(0.7, 0.7, 0.7))
joint = sphere(pos=vector(0, 1, 0), radius=0.1, color=vector(0, 0.5, 1))
arm = cylinder(pos=joint.pos, axis=vector(1, 0, 0), radius=0.05, color=vector
    (1, 0, 0))

# Target Position
target = sphere(pos=vector(2, 1.5, 0), radius=0.1, color=vector(0, 1, 0))

# System Parameters
J = 1.0 # Moment of inertia
b = 0.5 # Damping coefficient
k = 2.0 # Spring constant

# Gains (adjustable via sliders)
K_p = 1.0
K_d = 0.5

# Feedback Loop Variables
current_pos = joint.pos + arm.axis
previous_error = vector(0, 0, 0)
dt = 0.01 # Time step
tolerance = 0.05 # Tolerance for reaching the target

# Labels for displaying data
current_label = label(pos=vector(-3, 3, 0), text="Current Position: ", height
    =12, color=vector(0, 0, 0))
target_label = label(pos=vector(-3, 2.5, 0), text="Target Position: ", height
    =12, color=vector(0, 0, 0))
stability_label = label(pos=vector(-3, 2, 0), text="Stability: ", height=12,
    color=vector(0, 0, 0))

# Interactive Target Movement
def set_new_target(evt):
    global target, start_time, is_reaching
    target.pos = evt.pos # Update target position to the clicked position
    start_time = time.time() # Reset the timer
    is_reaching = True # Start tracking the reaching process
```

```python

scene.bind('click', set_new_target)

# Sliders for K_p and K_d
def update_kp(s):
    global K_p
    K_p = s.value

def update_kd(s):
    global K_d
    K_d = s.value

kp_slider = slider(min=0.1, max=5, value=K_p, length=300, bind=update_kp, right=15)
kp_text = wtext(text=f"K_p: {K_p:.2f}")

kd_slider = slider(min=0.1, max=5, value=K_d, length=300, bind=update_kd, right=15)
kd_text = wtext(text=f"K_d: {K_d:.2f}")

# Table to Record Results
results = []
result_label = label(pos=vector(-3, -3, 0), text="Results Table:\n", height=12, color=vector(0, 0, 0))

def update_results_table():
    table_text = "Results Table:\n"
    for i, (kp, kd, time_taken) in enumerate(results):
        table_text += f"{i+1}. K_p: {kp:.2f}, K_d: {kd:.2f}, Time: {time_taken:.2f}s\n"
    result_label.text = table_text

def clear_results():
    global results
    results = []
    update_results_table()

clear_button = button(text="Clear Table", pos=scene.title_anchor, bind=lambda _: clear_results())

# Simulation Variables
start_time = None
is_reaching = False

# Simulation Loop
while True:
    rate(100) # Simulation speed
    kp_text.text = f"K_p: {K_p:.2f}"
    kd_text.text = f"K_d: {K_d:.2f}"

    # Calculate Error
```

```
    error = target.pos - current_pos
    derivative = (error - previous_error) / dt

    # Laplace Transfer Function Control Signal
    control_signal = (1 / (J * dt**2 + (b + K_d) * dt + (k + K_p))) * error

    # Update Arm Position
    arm.axis += control_signal * dt
    current_pos = joint.pos + arm.axis

    # Stability Analysis
    discriminant = (b + K_d)**2 - 4 * J * (k + K_p)
    if discriminant > 0:
        stability_type = "Overdamped"
        pole1 = (-b - K_d + math.sqrt(discriminant)) / (2 * J)
        pole2 = (-b - K_d - math.sqrt(discriminant)) / (2 * J)
    elif discriminant == 0:
        stability_type = "Critically Damped"
        pole1 = pole2 = (-b - K_d) / (2 * J)
    else:
        stability_type = "Underdamped"
        real_part = (-b - K_d) / (2 * J)
        imaginary_part = math.sqrt(-discriminant) / (2 * J)
        pole1 = f"{real_part:.2f} + {imaginary_part:.2f}i"
        pole2 = f"{real_part:.2f} - {imaginary_part:.2f}i"

    # Update Labels
    current_label.text = f"Current Position: {current_pos.x:.2f}, {current_pos.
        y:.2f}, {current_pos.z:.2f}"
    target_label.text = f"Target Position: {target.pos.x:.2f}, {target.pos.y:.2
        f}, {target.pos.z:.2f}"
    stability_label.text = f"Stability: {stability_type}, Poles: {pole1}, {
        pole2}"

    # Check if the target is reached
    if is_reaching and error.mag < tolerance:
        time_taken = time.time() - start_time
        results.append((K_p, K_d, time_taken)) # Record the result
        update_results_table() # Update the table
        is_reaching = False # Stop tracking this event

    previous_error = error
```

7.5 Code for the Graph

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import TransferFunction, step
from ipywidgets import interactive


def step_response(J, b, k, Kp, Kd):
    """
    Plots the step response of a robotic arm control system using adjustable
        parameters.

    Parameters:
    J - Moment of inertia
    b - Damping coefficient
    k - Spring constant
    Kp - Proportional gain
    Kd - Derivative gain
    """


    num = [1]
    den = [J, b + Kd, k + Kp]

    try:

        system = TransferFunction(num, den)


        time, response = step(system)


        plt.figure(figsize=(8, 6))
        plt.plot(time, response, label=f"J={J}, b={b}, k={k}, Kp={Kp}, Kd={Kd}"
            )
        plt.title("Step Response of Robotic Arm with PID Control")
        plt.xlabel("Time (seconds)")
        plt.ylabel("Angular Position")
        plt.grid()
        plt.legend()
        plt.show()
    except Exception as e:
        print(f"Error in generating transfer function: {e}")


interactive_plot = interactive(
    step_response,
    J=(0.1, 5.0, 0.1),
    b=(0.1, 10.0, 0.1),
    k=(0.1, 10.0, 0.1),
```

```
    Kp=(1.0, 50.0, 1.0),
    Kd=(0.1, 10.0, 0.1)
)


interactive_plot
```

8 Conclusion

Laplace transformations are an indispensable tool in the analysis and design of feedback control systems, particularly in AI-based applications. They allow for the simplification of complex differential equations, aid in system stability analysis, and provide the necessary framework for optimizing control strategies.

9 References

- MIT Open Course Ware – Feedback Control Systems
- Khan Academy - Laplace Transforms
- Control Tutorials for Vpython
- System Poles and Zeroes - Libre office texts / Transfer function

10 Bibliography

- Ogata, K. (2010). *Modern Control Engineering* (5th ed.). Prentice Hall.
- Nise, N. S. (2011). *Control Systems Engineering* (6th ed.). Wiley.
- Dorf, R. C., & Bishop, R. H. (2017). *Modern Control Systems* (13th ed.). Pearson.

11 Signature

Aarnya Jain 2401030209
Vansh Garg 2401030202
Amolik Agarwal 2401030193
Paras Mehta 2401030210
Yatin Khatri 2401030214