Indian Institute of Technology, Gandhinagar

Ball & Beam Part-01

ES245 Control Systems

Team Vadapav - Group 04

Authors	*Roll Number:*
Deepak soni	22110068
Shaurykumar Patel	22110241
Piyush singh	22110253
Sridhar Singh	22110257

October 20, 2024

Contents

1 Introduction

The ball and beam system is a classical control problem that demonstrates the concepts of feedback control and system dynamics. In this system, a ball is placed on a beam and is allowed to roll freely under the influence of gravity along the beams length. The beam's angle can be controlled by a servo motor, which adjusts the inclination of the beam through a lever arm mechanism. The primary objective of this project is to design a controller that can stabilize the ball at a desired position on the beam by manipulating the servo motor's angle.

This project challenges students to model the dynamics of the ball and beam system, analyze its stability, and implement different control strategies. Given the non-linear nature of the system, the behavior of the ball is influenced by factors such as its mass, radius, moment of inertia, and the geometry of the beam. The system's complexity requires a step-by-step approach to linearization, transfer function analysis, and the design of an effective feedback controller.

The ultimate goal is to use a proportional-integral-derivative (PID) controller to manipulate the balls position, minimizing overshoot, reducing settling time, and ensuring system stability. By simulating the system in MATLAB and Simulink, and later constructing a physical prototype, students will explore real-world challenges in control systems, such as tuning PID gains and achieving the desired performance criteria.

This project is not only an exploration of control theory but also a practical exercise in system design, modeling, and implementation. The final deliverable will include a technical report, CAD drawings, programming scripts, and a physical system capable of controlling the balls position on the beam using a PID controller.

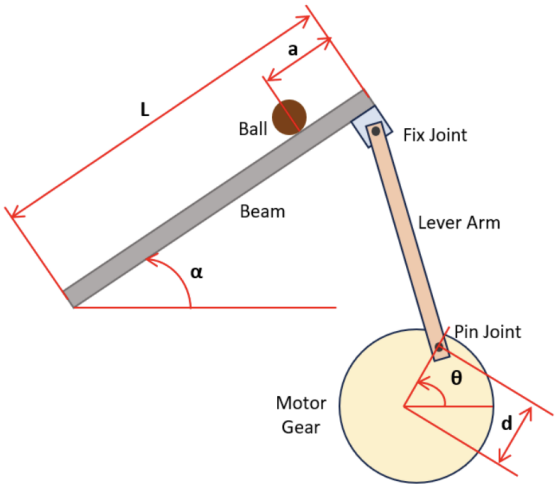


Figure 1: Ball and Beam schematic

2 Setup Overview

2.1 List of Materials

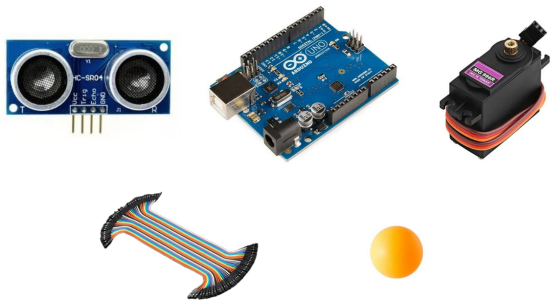


Figure 2: Materials

2.2 Hardware Overview

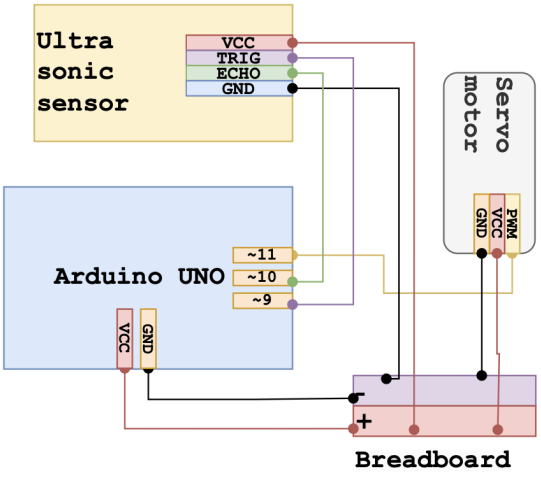


Figure 3: Circuit diagram

This was the electrical/hardware setup used in our physical ball and beam model.

- Ultrasonic sensor were used for measuring distance and based on changing the level of beam.
- Servo motor were used for tilting beam so that ball can be balanced at a certain point.
- Arduino UNO is used for fetching code and giving the input to the servo motor based on ultrasonic sensor

3 Essential Background

As shown in the figure below, a ball is placed on a beam, where it rolls under the influence of gravity. The ball has one degree of freedom along the length of the beam. The beam's angle is adjusted using a lever arm connected to the beam at one end and to a servo motor gear at the other. As the servo gear rotates by an angle θ about its axis, the lever arm angle changes by α from the horizontal.
Where :

- m is the mass of ball (in Kg)
- R is the radius of ball (in m)
- α is the angle of beam (in rad)
- J is the moment of inertia of ball (in $Kg{-}m^2$)
- J_m combined moment of inertia of ball and beam (in $Kg - m^2$)

Assumptions : The ball remains in contact with the beam at all times. The ball performs pure rolling motion (rolling without slipping).

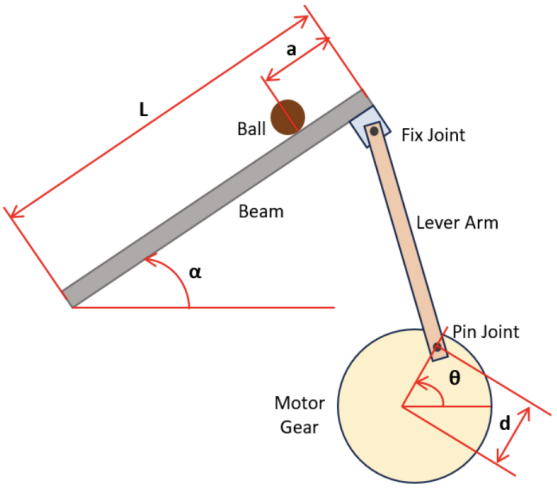


Figure 4: Ball and Beam Schematic

$$\text{Total Potential Energy(V)} = mgR\sin\alpha + \frac{1}{2}MgL\sin\alpha$$

$$\text{Total Kinetic Energy(T)} = \frac{1}{2}m(\dot{r}^2 + r^2\dot{\alpha}) + \frac{1}{2}J(\frac{\dot{r}}{R})^2 + \frac{1}{2}J_m(\dot{\alpha})^2$$

$$\text{Lagrangian (L)} = \text{T-V}$$

$$L = \frac{1}{2}m(\dot{r}^2 + r^2\dot{\alpha}) + \frac{1}{2}J(\frac{\dot{r}}{R})^2 + \frac{1}{2}J_m(\dot{\alpha})^2 - mgR\sin\alpha + \frac{1}{2}MgL\sin\alpha$$

$$\text{Using } \frac{d}{dt}(\frac{\partial L}{\partial \dot{r}}) - \frac{\partial L}{\partial r} = 0$$

$$(\frac{J}{R^2} + m)\ddot{r} + mg\sin\alpha - m\dot{r}\alpha^2 = 0$$

Linearization of this equation about the beam angle, $\alpha = 0$, gives us the following linear approximation of the system:

$$\left(\frac{J}{R^2} + m\right)\ddot{r} = -mg\alpha$$

Substituting $\alpha = \frac{d}{L}\theta$ in above equation, we get

$$\left(\frac{J}{R^2} + m\right)R(s)s^2 = -mg\frac{d}{L}\Theta(s)$$

Taking the Laplace transform of the equation above, the following equation is found:

$$\left(\frac{J}{R^2}+m\right)R(s)s^2 = -mg\frac{d}{L}\Theta(s)$$

transfer function from the gear angle ($\Theta(s)$) to the ball position ($R(s)$).

$$P(s) = \frac{R(s)}{\Theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2}+m\right)}\frac{1}{s^2} \qquad [\frac{m}{rad}]$$

3.1 Task 1.1

Write the system dynamics equations.

$$(\frac{J}{R^2}+m)\ddot{r} + mg\sin\alpha - m\dot{r}\alpha^2 = 0$$

3.2 Task 1.2

Write the system's transfer function between motor gear angle θ and ball position (a).

$$P(S) = \frac{R(S)}{C(S)} = -\frac{mgd}{L(\frac{J}{R^2}+m)}\frac{1}{S^2}$$

3.3 Task 1.3

Linearlize the system equations and write in the state space form.

$$\left(\frac{J}{R^2}+m\right)\ddot{r} = -mg\frac{d}{L}\theta$$

The linearized system equations can also be represented in state-space form.This can be done by selecting the ball's position (r) and velocity ($\dot{r}$) as the state variable and the gear angle (θ) as the input. The state-space representation is shown below:

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{mgd}{L\left(\frac{J}{R^2}+m\right)} \end{bmatrix}\theta$$

4 System Analysis

Assume suitable values of the parameters in the transfer function, including the mass of the ball (m), radius of the ball (R), lever arm offset (d), length of the beam (L), balls moment of inertia (J), balls position on beam (a), beam angle, and servo gear angle.

4.1 Task 2.1

Perform the following analysis using MATLAB.

- Plot poles/zeros

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

pzmap(P_ball)
```

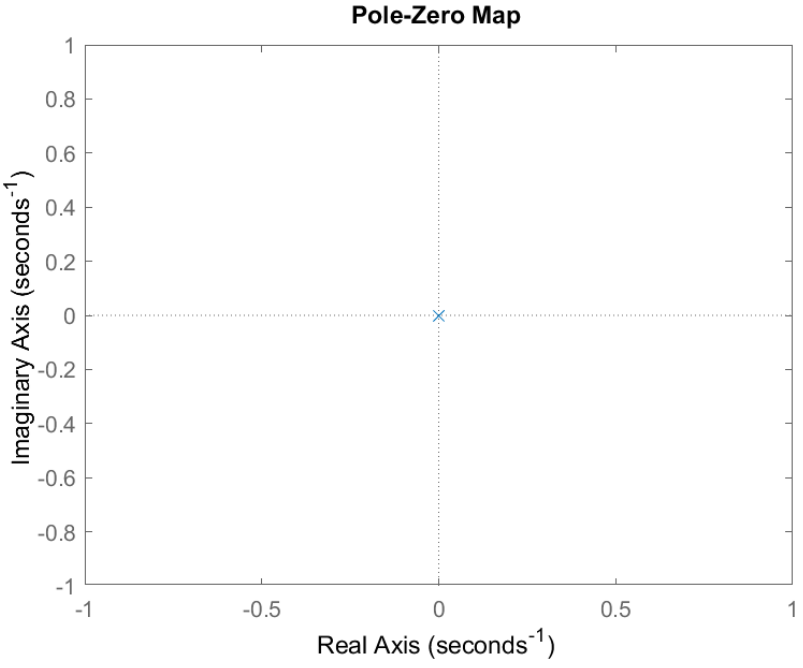


Figure 5: Pole Zero Map

- Plot open loop step response (servo gear angle in 1 radian step)

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

% Plot the step response
figure;

step(P_ball);

grid on;
title('Step Response of the Ball System');
xlabel('Time ');
ylabel('Response (Position)');
```

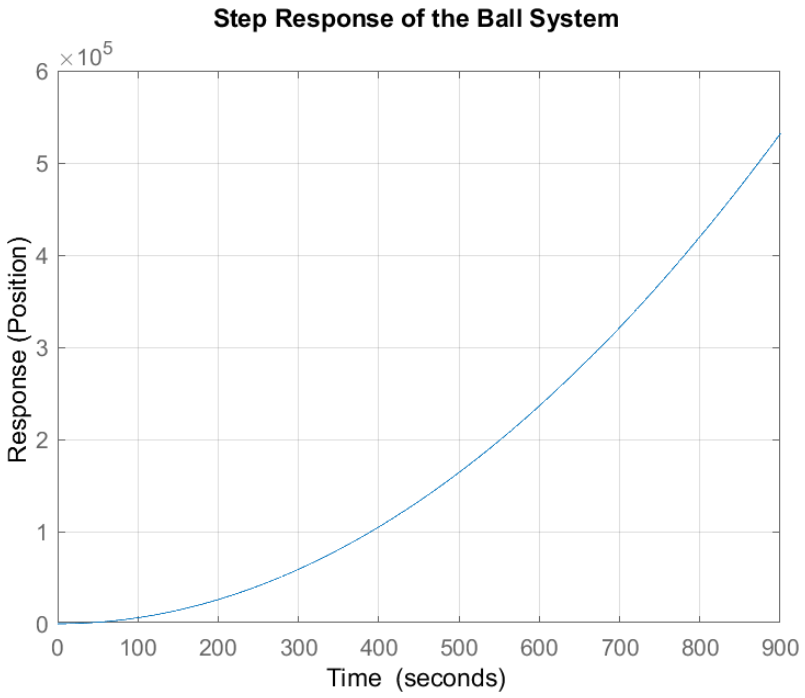


Figure 6: Open Loop Step Response

4.2 Task 2.2

Identify the type of the system and mention your observations from the poles/zeros plot and the open loop response of the system.

- The system has two poles at the origin, indicating it is a double integrator.
- The system is marginally stable but leads to unbounded behavior (position keeps increasing indefinitely) in response to a step input.
- The quadratic increase in the step response is typical of a double integrator, which leads to unbounded motion over time.

5 PID Control

5.1 Task 3.1

Design a controller with unity feedback and plot the performance of the following controllers for varying gains:

- Proportional controller

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

% Defining Kp ;
Kp = 0.5;
C = pid(Kp);
sys_cl=feedback(C*P_ball,1);

% Plotting response
[y,t] = step(0.25*sys_cl);
figure;
plot(t,y,'b','LineWidth',1.5);
grid;
title('Proportional Controller with Kp = 0.5');
xlabel('Time(seconds)');
ylabel('Amplitude');
axis([0 70 0 0.5])
```

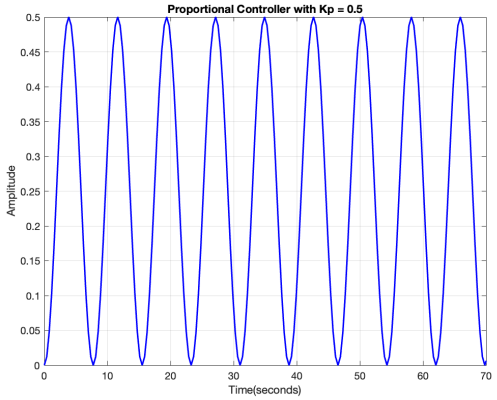


Figure 7: Step Response of the Proportional Controller

- Proportional-derivative controller

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

% Defining Kp and Kd
Kp = 10;
Kd = 10;
C = pid(Kp,0,Kd);
sys_cl=feedback(C*P_ball,1);

% Plotting Response
[y,t] = step(0.25*sys_cl);
figure;
plot(t,y,'b','LineWidth',1.5);
grid;
title('PD Controller with Kp = 10, Kd = 10');
xlabel('Time(seconds)');
ylabel('Amplitude');
t=0:0.01:5;
```

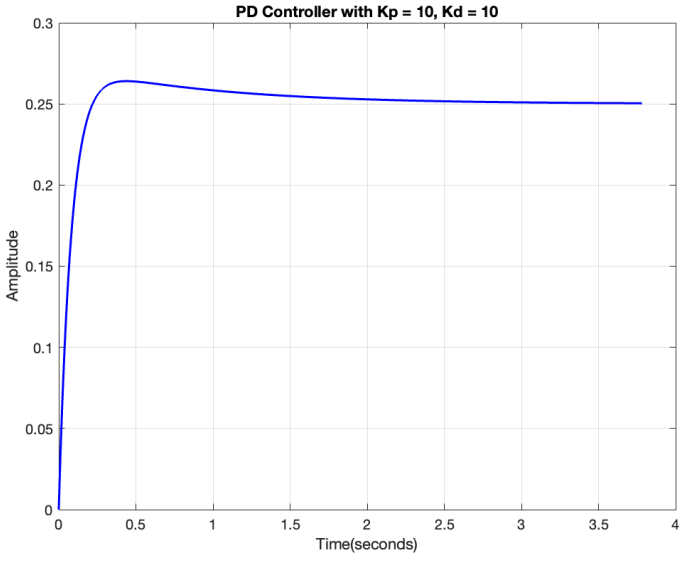


Figure 8: Step Response of the PD Controller

- Proportional-integral-derivative controller

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

% Defining Kp, Ki & Kd
Kp = 10;
Ki = 10;
Kd = 5;
C = pid(Kp,Ki,Kd);
sys_cl=feedback(C*P_ball,1);

% Plotting Response
[y,t] = step(0.25*sys_cl);
figure;
plot(t,y,'b','LineWidth',1.5);
grid;
title('PID Controller with Kp = 10, Ki = 10, Kd = 5')
   ;
xlabel('Time(seconds)');
ylabel('Amplitude');
t=0:0.01:5;
```

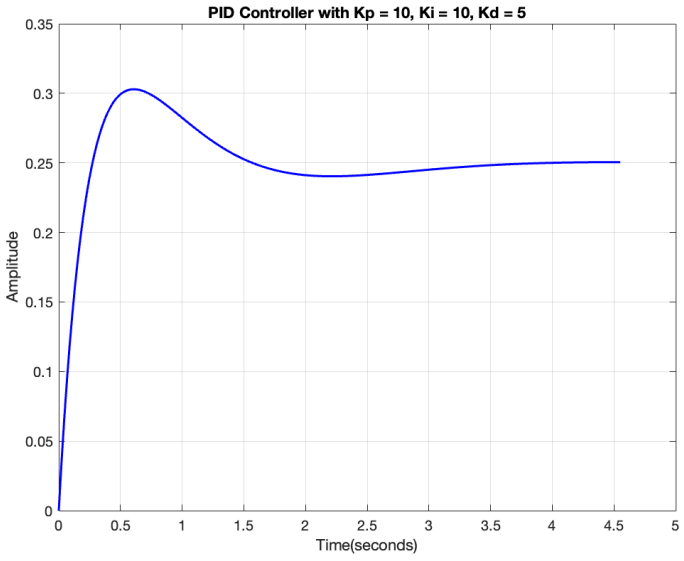


Figure 9: Step Response of the PID Controller

5.2 Task 3.2

Mention your observations on all three controller types and how they affect transient response, steady-state response, and set point error.

- **Proportional Controller** Increasing Kp will Fast response, but may introduce overshoot or oscillation. A proportional controller cannot eliminate steady-state error entirely.The higher the proportional gain, the smaller the steady-state error, but it won't be zero.Proportional controllers reduce the set point error but don't completely eliminate it.
- **Proportional + Derivative Controllers** Introducing Kd and on increasing it will reduce overshoot and making the system's response faster and smoother. A PD controller still cannot eliminate steady-state error, so expect a small residual error in steady-state. The error should reduce faster compared to a proportional-only controller, and the response will settle more smoothly.
- **PID Controller** The addition of the integral component helps improve the transient response, but it can also introduce overshoot and oscillation, particularly if the integral gain (Ki) is too high. The PID controller eliminates steady-state error through its integral action. It helps to eliminates set point error with smooth transitions

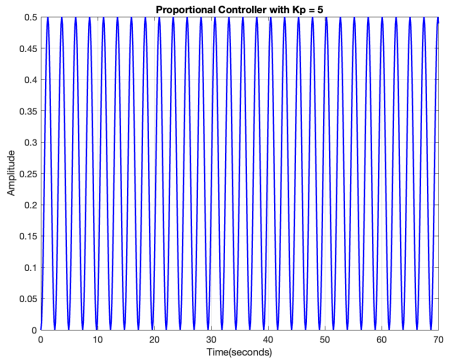


Figure 10: Kp = 5

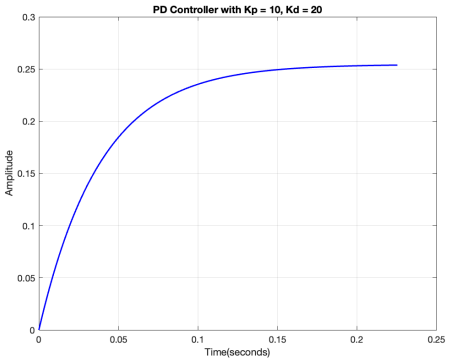


Figure 11: Kp = 10, Kd = 20

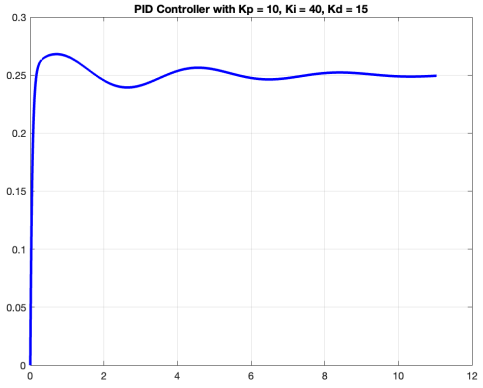


Figure 12: Kp =10, Ki =40 & Kd = 15

Figure 13: Response for different values of Kp, Ki and Kd.

5.3 Task 3.3

Design a controller to follow the following criteria - settling time less than 3 seconds and overshoot less than 5

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;

Kp = 10;
Ki = 20;
Kd = 15 ;
C = pid(Kp,Ki,Kd);
sys_cl = feedback(C*P_ball, 1);

% Plotting Response
[y, t] = step(0.25*sys_cl);
figure;
plot(t, y, 'b-', 'LineWidth', 2.5); % Thicker blue line
hold on;
grid on;
info = stepinfo(sys_cl);
fprintf('Maximum overshoot time : %.2f %%\n',info.
   Overshoot);
fprintf('Settling time : %.4f seconds\n',info.
   SettlingTime);
% Maximum Overshoot
overshoot_time = info.PeakTime; % Time at maximum peak
max_value = max(y);  % Maximum value of the step response
plot(overshoot_time, max_value, 'ro', 'MarkerSize', 8, '
   MarkerFaceColor', 'r'); % Red marker at peak
ylim([0 0.3])
% Settling Time
settling_value = y(find(t >= info.SettlingTime, 1));
plot(info.SettlingTime, settling_value, 'blacko', '
   MarkerSize', 8, 'MarkerFaceColor', 'black'); % Green
   marker at settling time


text(overshoot_time, max_value, sprintf('  Max Overshoot:
    %.2f%%', info.Overshoot), ...
    'VerticalAlignment', 'top', 'HorizontalAlignment', '
       left', ...
```

```
    'FontSize', 10,'Color', 'r');
text(info.SettlingTime, settling_value, sprintf('
   Settling Time: %.2f s', info.SettlingTime), ...
    'VerticalAlignment', 'top', 'HorizontalAlignment', '
       left', ...
    'FontSize', 10, 'Color', 'black');

xlabel('Time (seconds)', 'FontSize', 12, 'FontWeight', '
   bold');
ylabel('Amplitude', 'FontSize', 12, 'FontWeight', 'bold')
   ;
title('PID Controller with Kp = 10, Ki = 20, Kd = 15', '
   FontSize', 14, 'FontWeight', 'bold');
legend('Step Response', 'Max Overshoot', 'Settling Time',
    'Location', 'Best');
axis([0 max(t) 0 0.5]);
hold off;
ylim([0 0.3])
```

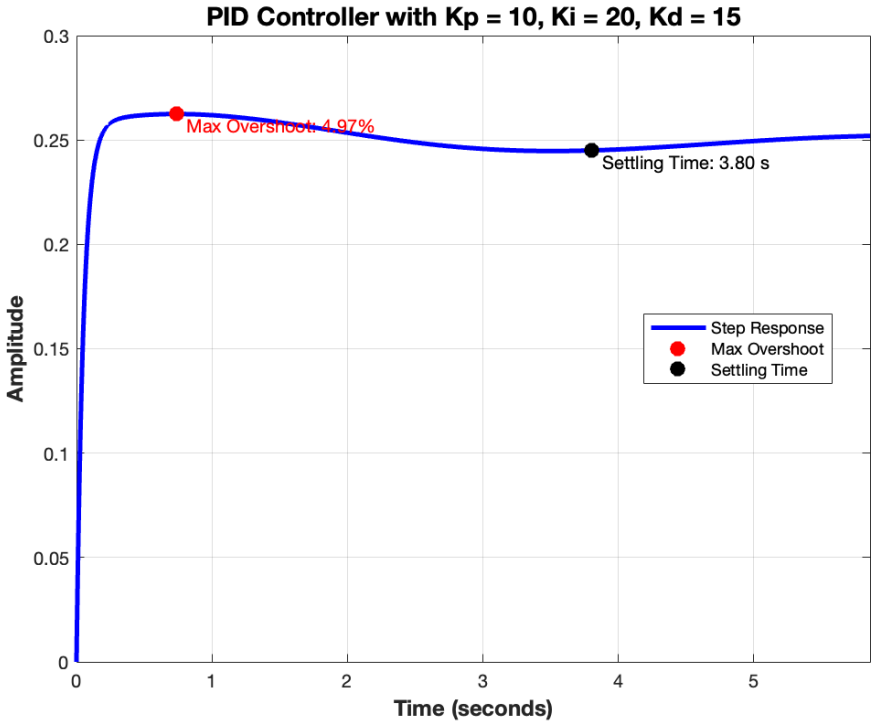


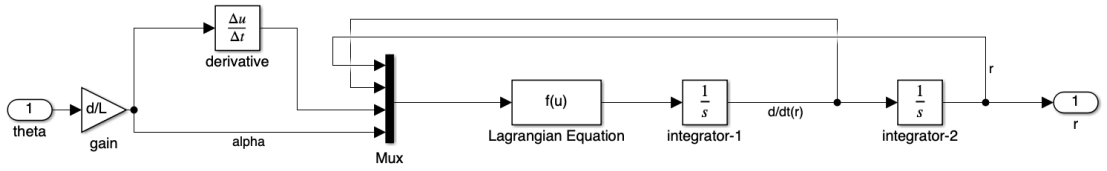
Figure 14: Step Response of the PID Controller

6 Simulation of ball and beam in MATLAB

6.1 Task 4.1

Build the ball beam model in Simulink and generate the system's open loop response.

To create the Ball-Beam system model in Simulink, two Integrator blocks were inserted and connected, with the line labeled "d/dt(r)." The output from the second Integrator, labeled "r," was connected to an Out1 block. A Mux block with four inputs was added, with the "d/dt(r)" and "r" signals connected to the second and first inputs, respectively.



An In block labeled "theta" was connected to a Gain block with a gain value of "d/L," and the output was connected to the third input of the Mux. A Derivative block was added, with its input tapped from the Gain block output and connected to the fourth Mux input.

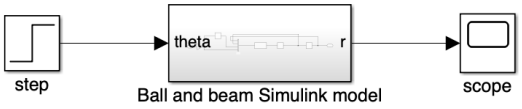


Figure 15: Ball beam model in Simulink

After running the simulation the below plot was generated in the scope.

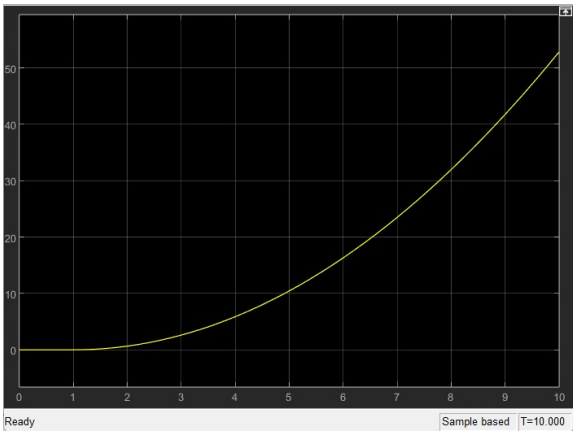


Figure 16: Open loop Response of the model from Simulink

6.2 Task 4.2

Linearize the model and design a compensator to meet the following design criteria - overshoot of less than 5% and settling time of less than 5 seconds. Generate the system's closed-loop response.
A first-order lead compensator C(s) can be designed using the root locus. A lead compensator in root locus form is given by

$$C(s) = K_c \frac{(s + z_0)}{(s + p_0)}$$

where, the magnitude of z_0 is less than the magnitude of p_0. To meet design criteria;

$$Mp = e^{-\zeta\pi/\sqrt{1-\zeta^2}} = 0.05, T_s = \frac{4}{\zeta\omega_n} = 5\text{seconds}$$

From the above formulae, we got $\zeta = 0.70$ *and* $\omega_n = 1.142 rad/s$

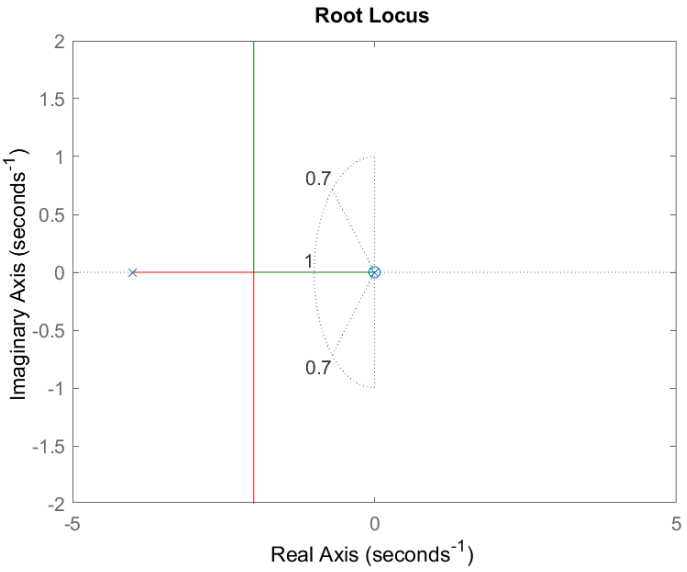


Figure 17: Root locus with design criteria.

The lead compensator controller for the model was added using the transfer function $\frac{s+0.001}{s+4}$ and gain of 5 after trying multiple values of z_0, p_0 and K to meet the given design criteria. We used MATLAB to plot the closed loop response to design the Lead compensator with the given criteria.

```
m = 0.0027;
R = 0.02;
g = -9.8;
L = 0.342;
d = 0.064123;
J = 4.32e-7;
s = tf('s');
P_ball = -m*g*d/L/(J/R^2+m)/s^2;
```

```
% Lead compensator design.
z0 = 0.001;
p0 = 4;
C = tf([1 z0], [1 p0]);

% Plot the root locus of the compensated system
figure;
rlocus(C*P_ball);
sgrid(0.7, 1.0);
axis([-5 5 -2 2]);

k = 5; % Adjusting gain to meet design criteria
sys_cl = feedback(k*C*P_ball, 1);
t = 0:0.01:5;
figure;
step(0.25*sys_cl, t);
grid on;
title("Closed Loop Response");

% Get step response performance metrics
info = stepinfo(sys_cl);
fprintf('Maximum overshoot time : %.2f %%\n', info.
   Overshoot);
fprintf('Settling time : %.4f seconds\n',info.
   SettlingTime);
```

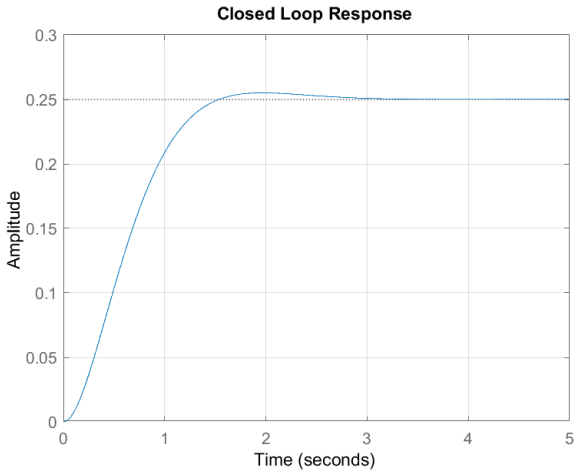


Figure 18: Closed loop Response of lead compensator with Overshoot % = 2.03% and T_s = 2.04 seconds

6.3 Task 4.3

Develop a Simscape model for the ball beam system. The ball-and-beam model in Simscape simulates how a ball moves along a beam, with the beam's rotation controlled to maintain the ball's position. Simscape Multibody is used to simulate rigid bodies, joints, and forces, providing realistic dynamics. The ball moves along the beam governed by rotational dynamics, while the beam pivots around a revolute joint. A gear mechanism adjusts the beams angle, and subsystems like the beam, ball, and gear are modularized for simplicity. Revolute joints allow the beam's rotation, while prismatic constraints restrict the ball's movement along the beam. PID controller adjusts the beams angle based on feedback. The model also includes physical constraints, unit conversions, and real-time visualization, capturing the systems unstable nature and allowing for optimization and testing of control strategies.

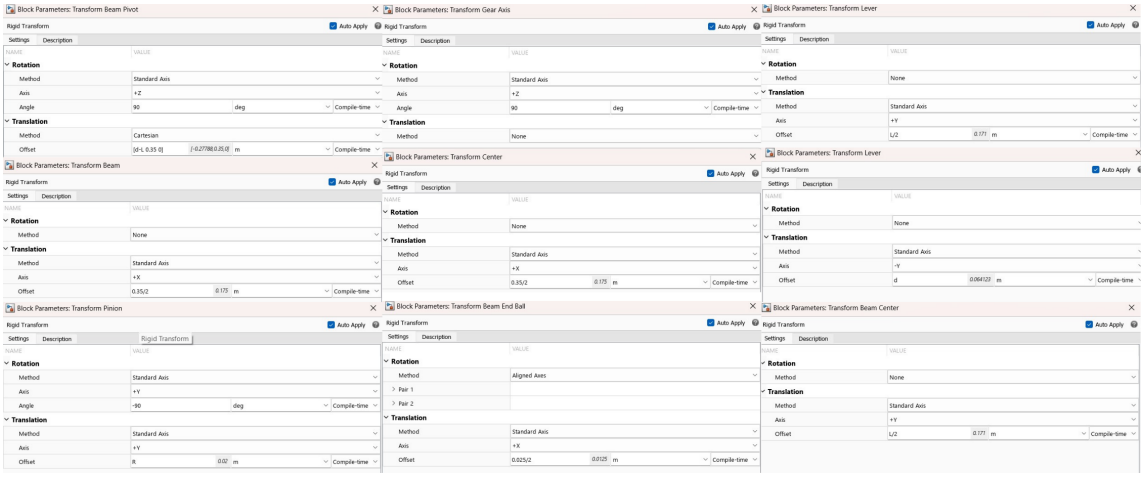


Figure 19: Transform Constraints

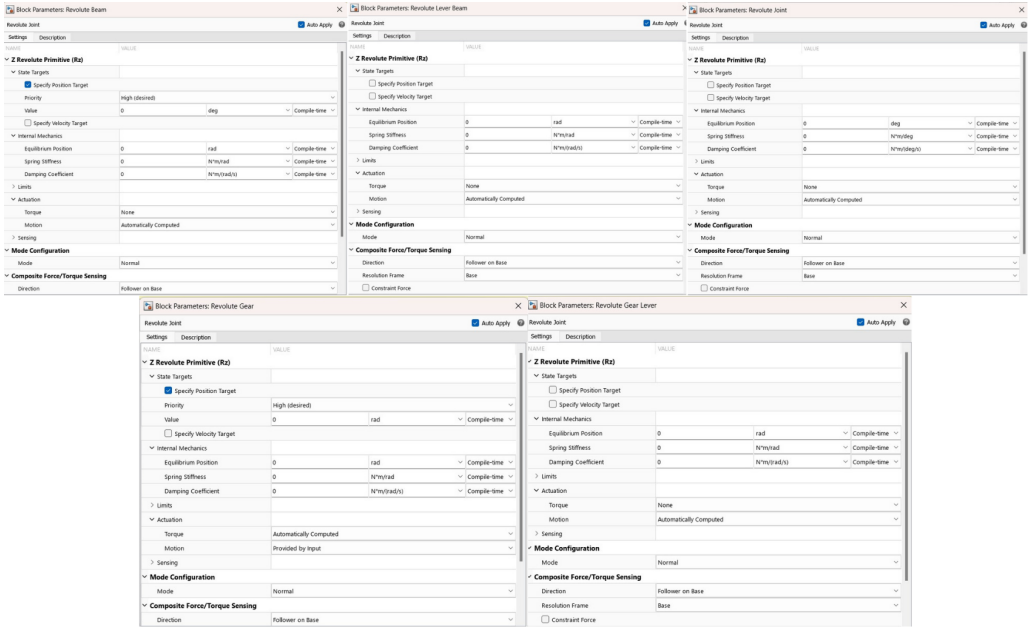


Figure 20: Revolute Constraints

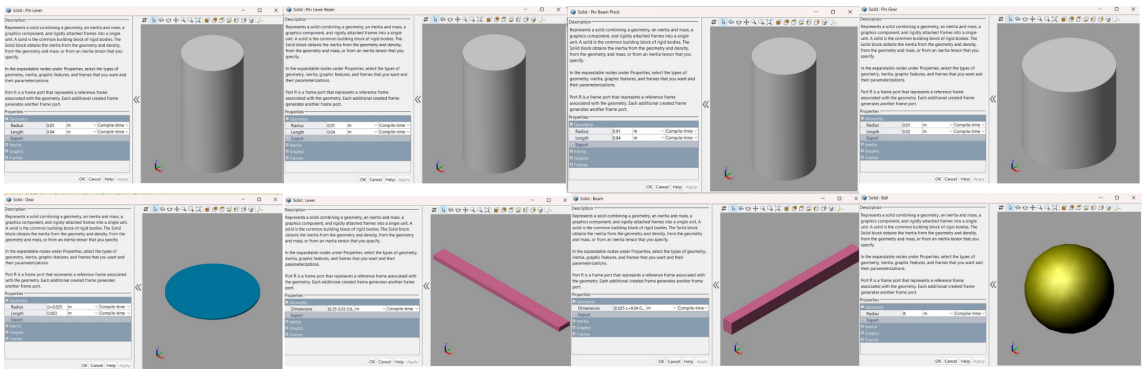


Figure 21: Parts Constraints

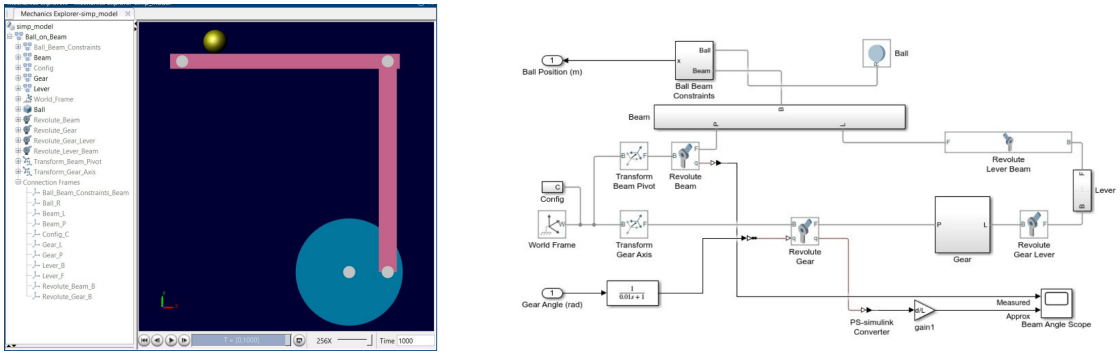
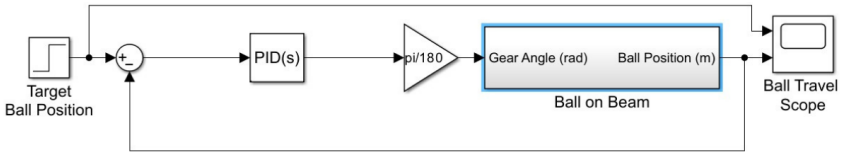


Figure 22: Simscape Model



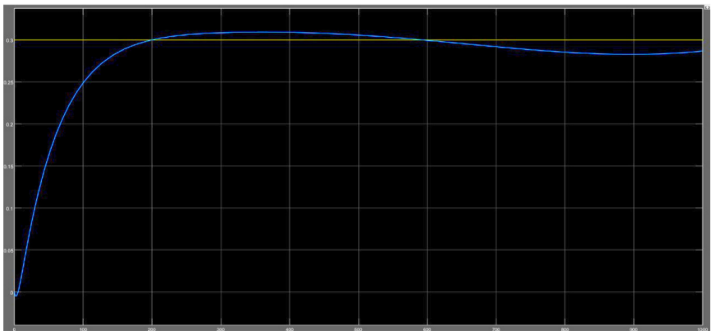


Figure 23: PID Control on Simscape

7 Physical System

7.1 Task 5.1

Develop a CAD model of the system and build the physical system.

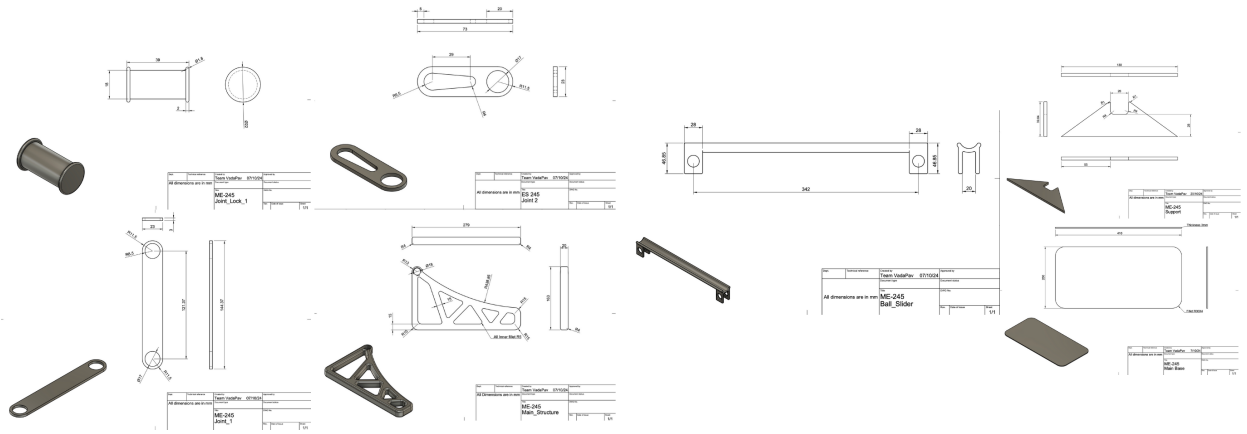


Figure 24: Individual Parts with Dimensions

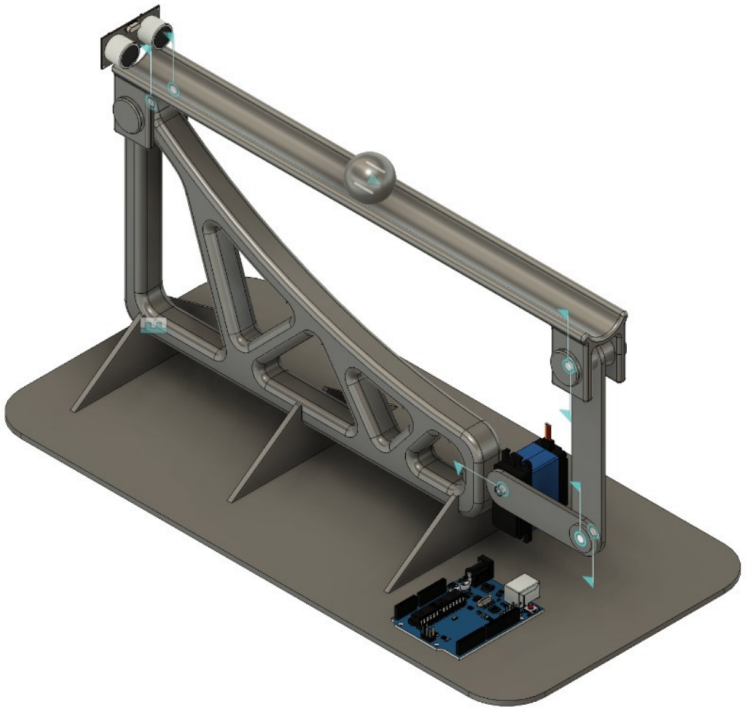


Figure 25: CAD Model

7.2 Task 5.2

Design a PID controller to stabilize the ball on the beam at the given distance from the end. Comment on your observations and challenges in implementing the PID control.

- **Increasing (Proportional Gain) Behavior:** Causes oscillatory motion, leading to a "to and fro" motion from one end of the beam to the other. Explanation: Higher kp increases responsiveness but also results in excessive overshoot, preventing the system from stabilizing around the target position.
- **Increasing (Integral Gain) Behavior:** Increases motor speed, making it challenging to attain equilibrium. Explanation: A higher Ki causes the control output to become excessively large, driving the motor too quickly and leading to instability and overshoot.
- **Combined Effects and Challenges:** The combination of increased Kp and Ki results in oscillations and high-speed corrections, creating a feedback loop that prevents stabilization. Damping through Kd may be necessary to mitigate these effects.
- **Recommendations**
  1. Balanced Tuning: Gradually reduce K_p and K_i to stabilize the system before incrementally increasing them.
  2. Introduce K_d: Adjust K_d to add damping and reduce oscillations.
  3. Integral Windup Protection: Implement strategies to prevent integral windup.
  4. Simulation Testing: Use simulations to evaluate changes before applying them to the actual system.

Figure 26: CAD Model

Watch Video: PID Control

8 Conclusion

The Ball and Beam system is a fascinating control problem that demonstrates fundamental concepts in control theory, particularly when balancing a ball on a beam. The challenge lies in designing a control system that can stabilize the inherently unstable system. Among the various control strategies, the PID controller plays a crucial role, with the proportional (K_p), integral (K_i), and derivative (K_d) gains being critical parameters that must be carefully tuned. Tuning these parameters correctly is essential for ensuring system stability, fast response, and minimizing overshoot and steady-state error. The proportional term influences the system's reaction to current errors, the integral term helps eliminate steady-state errors, and the derivative term predicts future errors based on the rate of change. An improper selection of these values can lead to poor performance, such as slow response, oscillations, or even instability. Through careful tuning and implementation of the PID controller, the Ball and Beam system can be successfully stabilized, making it an excellent demonstration of the power and challenges involved in real-time control systems.

9 Acknowledgement

We would like to thank Professor Vineet Vashista for his valuable guidance and support throughout this project. His knowledge made a huge difference in our understanding and learning. We are also grateful to our Teaching Assistants, Rajdeep singh devra and Jenish Chauhan, for their help and dedication during our regular review meetings. Their support was key in helping us overcome many challenges.

References

[1] https://ctms.engin.umich.edu/CTMS/?example=BallBeam&section=SystemModeling

[2] https://pubs.sciepub.com/acis/3/1/1/

[3] https://www.tandfonline.com/doi/full/10.1080/21642583.2019.1575297