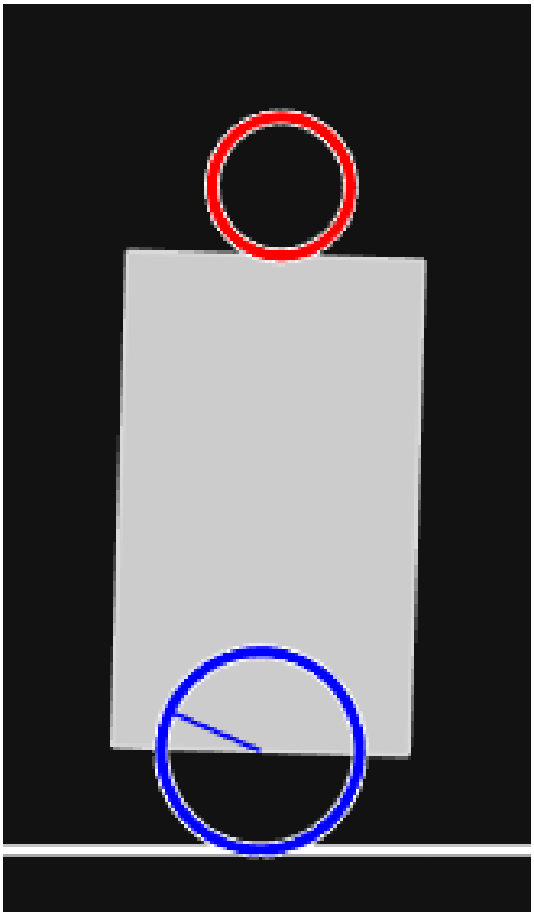
DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

Control Systems Project: Self Balancing Robot



A Project Report submitted by

Adwait Thosare
Roll No: ME23B123

ME4010 – Control Systems

December 2025

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Abstract

This project explores modeling, simulation, and control of a self-balancing robot designed to stabilize a ball on its head. MATLAB and Simulink are used to develop mathematical models, design controllers (PID and full-state feedback), and implement observers to handle measurement noise. The report presents comparative analyses of control schemes and quantifies their performance metrics.

Contents

Chapter 1

Introduction

Balancing an inverted pendulum is a classic benchmark problem in control engineering due to its nonlinear, underactuated, and inherently unstable nature. In this project, the challenge is extended: the self-balancing robot must remain upright while also keeping a free-rolling ball balanced on its head, creating a tightly coupled multi-instability problem that reacts quickly to disturbances.

Using only the robot's angular orientation from an IMU and wheel measurements using an encoder, the system must be modeled from first principles, linearized, and controlled. This project involves designing and comparing PID and full-state feedback controllers, developing full and reduced-order observers, and evaluating their performance under realistic sensor noise using filtering and spectral analysis. Overall, the work connects core theoretical concepts with practical issues such as noise, robustness, and implementation in MATLAB and Simulink.

Chapter 2

System Modelling

2.1 System Description

The system consists of a self-balancing robot modeled as an inverted pendulum mounted on a cart, with a free-rolling ball placed on top of the pendulum. The cart moves horizontally and is actuated by a DC motor; the control input is the torque applied to the wheels. Motion is restricted to a two-dimensional plane.

The robot measures only its angular orientation using an IMU, and wheel orientation may be inferred through an encoder if required. These measurements form the basis for estimating the remaining system states.

The addition of the ball introduces an extra degree of freedom and a second unstable mode. Even if the pendulum is held upright, the ball can accelerate along the pendulum unless corrective motion is continuously applied. This coupling makes the dynamics nonlinear and highly sensitive to disturbances.

A schematic representation of the system is shown in Fig. 2.1. The objective is to derive the governing nonlinear equations, express them in state-space form, and linearize them around the upright equilibrium to enable controller and observer design.

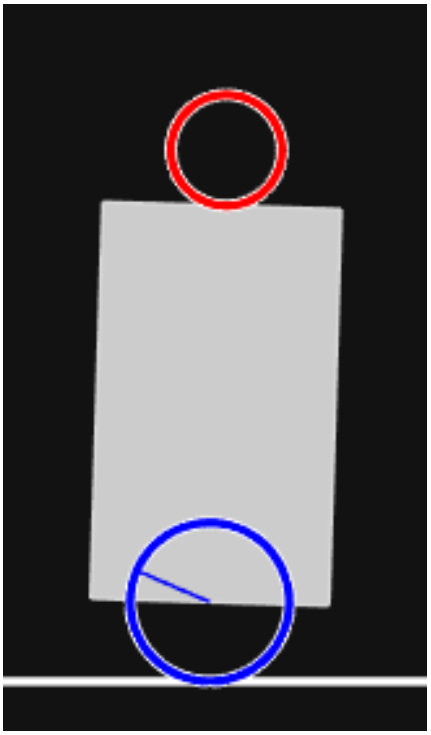


Figure 2.1: Schematic representation of the self-balancing robot system

2.2 Assumptions

- The ball rolls on the platform **without slipping**, ensuring a pure rolling motion.
- The ball **always remains in contact** with the platform; applied torque is not large enough to cause detachment.
- The entire system motion is **restricted to a 2D plane**.
- Both the **ball and the body (or the chassis)** are treated as **rigid bodies** with uniform mass distribution.
- The **pivot is frictionless**, and there is **no damping or air resistance** in the system.
- **No energy losses** occur due to rolling or mechanical resistance, the motion is idealized.

2.3 System Modeling and Derivation of Equations of Motion

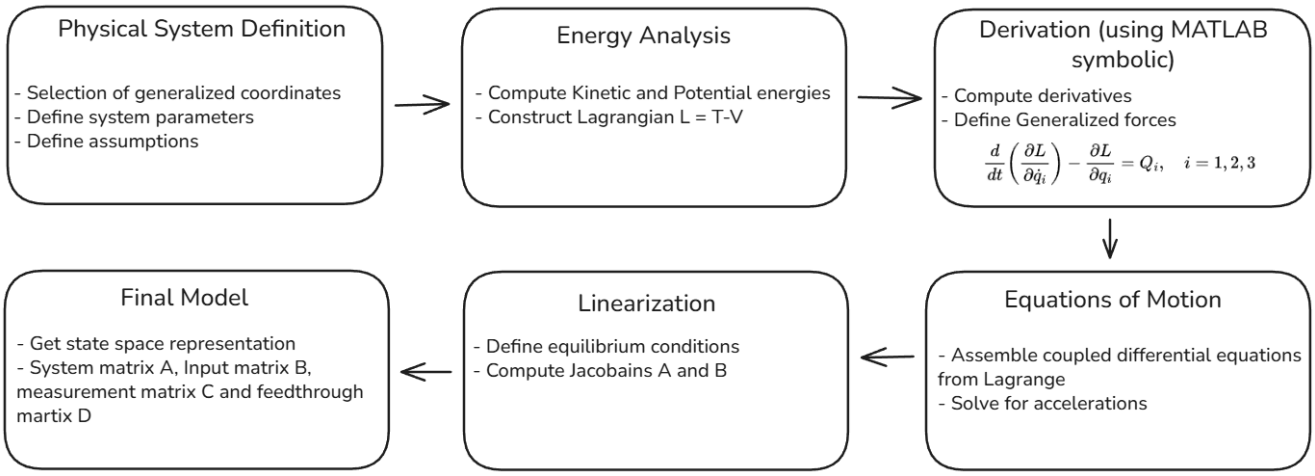


Figure 2.2: Flowchart illustrating the mathematical modeling methodology, detailing the progression from physical definition to the linearized state-space representation using Lagrangian mechanics.

2.3.1 Coordinate System and Generalized Coordinates

The following generalized coordinates are defined to describe the configuration:

- x: Horizontal displacement of the robot base.
- θ: Angular displacement of the body from the vertical (positive clockwise).
- x_b: Displacement of the ball relative to the center of the platform.

The generalized coordinate vector is therefore:

$$q = \begin{bmatrix} x \\ \theta \\ x_b \end{bmatrix}$$

Notation

x	: horizontal displacement of the robot base (inertial frame)
$\dot{x}$	: base linear velocity
θ	: body (beam) angle w.r.t. vertical (positive CCW)
$\dot{\theta}$	: angular velocity of the body
x_b	: ball displacement measured along the beam from its center
$\dot{x}_b$	: rate of change of x_b
R	: wheel radius
r	: ball radius
$M_{\text{wheel}}, I_{\text{wheel}}$	: mass and inertia of wheel assembly
$M_{\text{body}}, I_{\text{body}}$	: mass and inertia of the robot body
$m_{\text{ball}}, I_{\text{ball}}$	: mass and inertia of the ball
h	: height of the body.

2.3.2 Lagrangian and Equations of Motion

The dynamics of the system are derived using the Lagrangian formulation. The Lagrangian L is defined as the difference between the total kinetic and potential energies:

$$L = T - V$$

where $T = T_{\text{wheel}} + T_{\text{body}} + T_{\text{ball}}$

$$T = \frac{1}{2}M_{\text{wheel}}\dot{x}^2 + \frac{1}{2}I_{\text{wheel}}\left(\frac{\dot{x}}{R}\right)^2 + \frac{1}{2}M_{\text{body}}\left(\dot{x}_{\text{body}}^2 + \dot{y}_{\text{body}}^2\right) + \frac{1}{2}I_{\text{body}}\,\dot{\theta}^2 + \frac{1}{2}m_{\text{ball}}\left(\dot{x}_{\text{ball}}^2 + \dot{y}_{\text{ball}}^2\right) + \frac{1}{2}I_{\text{ball}}\left(\frac{\dot{x}_b}{r}\right)^2.$$

and $V = V_{\text{body}} + V_{\text{ball}}$

$$V_{\text{total}} = M_{\text{body}}g\left(\frac{h}{2}\cos\theta\right) + m_{\text{ball}}g\left[(h+r)\cos\theta - x_b\sin\theta\right].$$

The system is described by the generalized coordinates:

$$q = \begin{bmatrix} x \\ \theta \\ x_b \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{x}_b \end{bmatrix},$$

Lagrange's equations of motion are then written as:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i, \quad i = 1, 2, 3$$

where Q_i represent the generalized non-conservative forces associated with each coordinate.

For this system: - $Q_x = \tau_m / R_w$, the equivalent horizontal force on the wheel due to the applied motor torque τ_m. - $Q_\theta = -\tau_m$, as the motor torque acts in opposition on the body. - $Q_{x_b} = 0$, since the ball moves freely without external torque input.

2.4 Linearization and State Space Representation

The equations of motion derived from the Lagrangian are nonlinear due to the trigonometric dependencies on the tilt angle (θ) and the coupling between translational and rotational dynamics. To design and implement a feedback controller, it is necessary to linearize the system about an equilibrium point. Linearization simplifies the system to a form where linear control theory can be effectively applied, allowing us to study system stability, controllability, and response characteristics.

For the self-balancing robot, the equilibrium configuration corresponds to the upright position, where:

$$x = 0, \quad \dot{x} = 0, \quad \theta = 0, \quad \dot{\theta} = 0, \quad x_b = 0, \quad \dot{x_b} = 0, \quad \tau = 0$$

The nonlinear state-space equations are linearized using a first-order Taylor series expansion about this equilibrium point. Symbolic computation is used to obtain the Jacobian of the system dynamics with respect to the state and input vectors.

$$A = \left.\frac{\partial f}{\partial z}\right|_{(z_{eq}, u_{eq})}, \quad B = \left.\frac{\partial f}{\partial u}\right|_{(z_{eq}, u_{eq})}$$

where

$$z = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ x_b \\ \dot{x_b} \end{bmatrix}, \quad u = [\tau]$$

represent the state and input vectors, respectively.

Substituting the physical parameters into the derived symbolic Jacobians, the numerical state-space matrices for the upright equilibrium configuration are obtained as:

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8.6509 & 0 & -9.4168 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 64.2236 & 0 & 80.6014 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & -5.3536 & 0 & -16.5416 & 0 \end{bmatrix} \tag{2.1}$$

$$B = \begin{bmatrix} 0 \\ 0.6202 \\ 0 \\ -3.9120 \\ 0 \\ 0.6863 \end{bmatrix} \tag{2.2}$$

To complete the state-space representation, the output equation $y = Cz + Du$ is defined based on the available sensors. The system is equipped with wheel encoders to measure cart position (x) and an IMU to measure the chassis tilt angle (θ). Accordingly, the output matrix C is constructed to extract the first and third states:

$$y = \begin{bmatrix} \theta \\ x \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} z, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.3}$$

Model Validation properties

The feasibility of the proposed control strategies was verified by analyzing the structural properties of the linearized system. The controllability matrix $\mathcal{C}$ and observability matrix $\mathcal{O}$ were computed as:

$$\mathcal{C} = [B, AB, \ldots, A^{n-1}B], \quad \mathcal{O} = [C; CA; \ldots; CA^{n-1}] \tag{2.4}$$

Numerical analysis in MATLAB confirmed that both matrices are full rank:

$$\text{rank}(\mathcal{C}) = 6, \quad \text{rank}(\mathcal{O}) = 6 \tag{2.5}$$

The full rank of $\mathcal{C}$ confirms that the underactuated system is fully controllable via the single motor input. The full rank of $\mathcal{O}$ confirms that the unmeasured states (velocities and ball position) can be successfully reconstructed from the position and angle measurements, validating the approach for the Observer design in Chapter 4.

2.5 Simulation and Validation

2.5.1 Scenario 1: Equilibrium Stability Check

The system was initialized with zero initial conditions ($x_0 = \mathbf{0}$) and zero control input ($u = 0$). As shown in Figure 2.3, all state variables remained constant at zero. This confirms that the derived state-space model correctly identifies the vertical upright position as an equilibrium point.

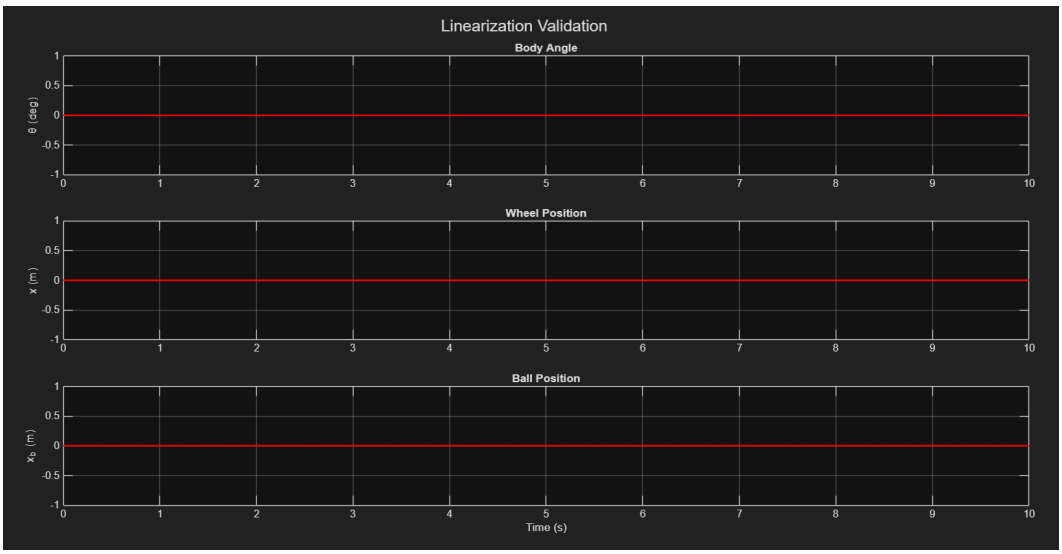


Figure 2.3: Open-Loop Response at Equilibrium

2.5.2 Scenario 2: Instability Verification (Free Fall)

To verify the open-loop instability inherent to the inverted pendulum, a small initial perturbation of $\theta_0 = 0.01$ rad was applied with no control input. Figure 2.4 illustrates the response. The chassis angle θ exhibits exponential divergence, which is consistent with the presence of a positive eigenvalue in the system matrix A ($\lambda > 0$). This confirms the model correctly captures the unstable dynamics of the pendulum.

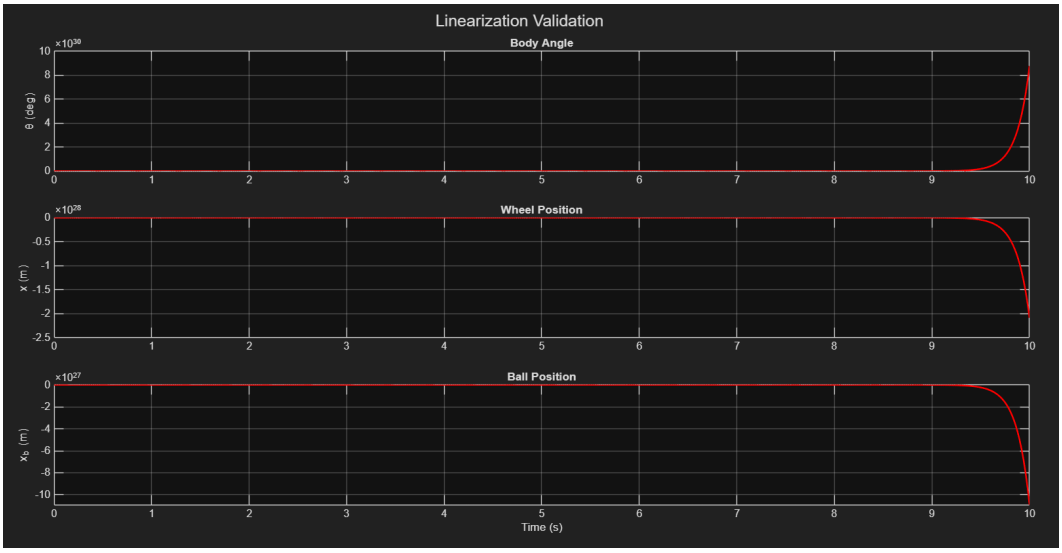


Figure 2.4: Open-Loop Response to Perturbation

2.5.3 Scenario 3: Actuator Coupling Verification

A step input of $u = 1$ was applied to the motors starting from the equilibrium position. Figure 2.5 demonstrates the physical coupling of the system:

1. The cart velocity $\dot{x}$ increases linearly in the positive direction.
2. The chassis angle θ initially tilts in the *negative* direction.

This negative tilt is the expected physical reaction torque (Newton's 3rd Law), as the wheels accelerate forward, the chassis experiences an opposing torque. This behavior confirms that the signs of the B matrix correctly map the physical relationship between motor voltage and system acceleration.

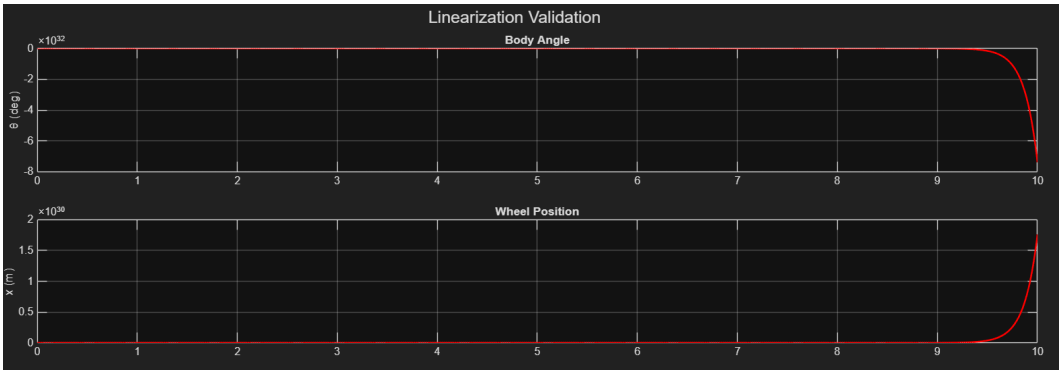


Figure 2.5: Open-Loop Step Response

Note: The simulation shows unbounded exponential growth (10^{30}). This is expected in a linear model which lacks the physical constraints (such as the floor or mechanical limits) present in the non-linear system.

Chapter 3

Controller Design

3.1 Transfer Function Derivation and Stability Analysis

Given (A, B, C, D), the transfer functions from the motor torque τ to the selected outputs are obtained using

$$G(s) = C(sI - A)^{-1}B + D. \tag{3.1}$$

The resulting transfer functions are:

$$G_\theta(s) = \frac{-3.912\,s^4 - 6.472 \times 10^{-24}\,s^3 - 9.395\,s^2 - 1.554 \times 10^{-23}\,s + 1.731 \times 10^{-15}}{s^6 + 6.661 \times 10^{-16}\,s^5 - 47.68\,s^4 - 3.895 \times 10^{-14}\,s^3 - 630.9\,s^2}.$$

$$G_{x_b}(s) = \frac{0.6202\,s^4 - 2.754 \times 10^{-16}\,s^3 - 2.194\,s^2 - 2.906 \times 10^{-15}\,s - 92.17}{s^6 + 6.661 \times 10^{-16}\,s^5 - 47.68\,s^4 - 3.895 \times 10^{-14}\,s^3 - 630.9\,s^2}.$$

Inspection of the characteristic polynomial reveals negative coefficients, mathematically confirming the open-loop instability observed in simulation.

3.2 PID Controller Design and Cascaded Loop Architecture

The system exhibits two strongly coupled unstable modes:

- Body angle θ must stabilize extremely quickly,
- The ball position x must be regulated more slowly, without destabilizing the body.

Therefore, a **cascaded PID architecture** is employed:

1. The **inner loop** stabilizes the body angle θ using a fast PID:

$$u_\theta(t) = K_{p\theta} e_\theta + K_{i\theta} \int e_\theta \, dt + K_{d\theta} \dot{e}_\theta$$

2. The **outer loop** regulates ball displacement x_b:

$$\theta_{\text{ref}}(t) = K_{px_b} e_{x_b} + K_{ix_b} \int e_{x_b} \, dt + K_{dx_b} \dot{e}_{x_b}$$

3. The inner loop uses θ_{ref} as a commanded body tilt, which indirectly moves the ball to the desired position.

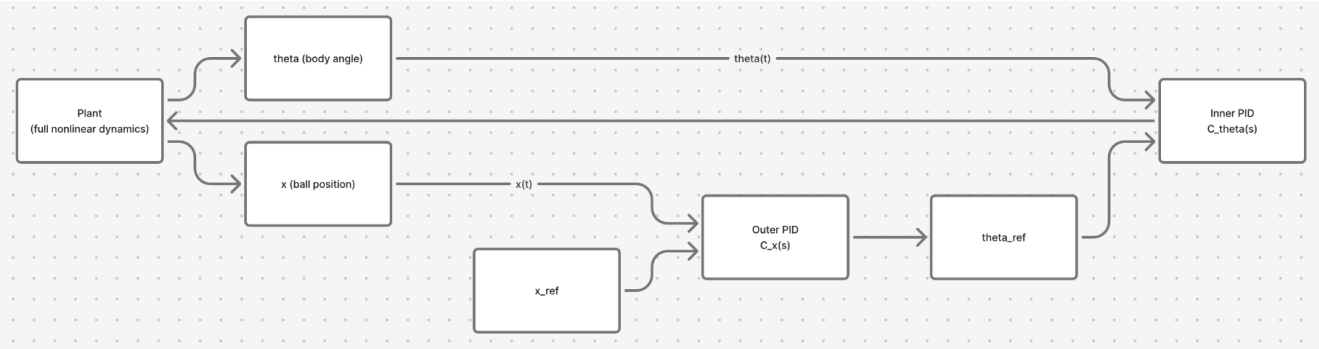


Figure 3.1: Cascaded PID control architecture. The outer loop compares the measured ball position $x_b(t)$ with the reference $x_{b,\text{ref}}$ and generates a commanded body angle θ_{ref}. The inner loop then drives the actual tilt $\theta(t)$ to track θ_{ref}. The resulting control input is applied to the full nonlinear plant.

3.2.1 Inner Loop Design (Body Angle Control)

The inner loop uses the transfer function $G_\theta(s)$ and a PID:

$$C_\theta(s) = K_{p\theta} + \frac{K_{i\theta}}{s} + K_{d\theta} s.$$

The gains were tuned to achieve a fast and well–damped response:

$$K_{p\theta} = 1200, \qquad K_{i\theta} = 1680, \qquad K_{d\theta} = 20.$$

The closed–loop transfer function is:

$$T_\theta(s) = \frac{C_\theta(s) G_\theta(s)}{1 + C_\theta(s) G_\theta(s)}.$$

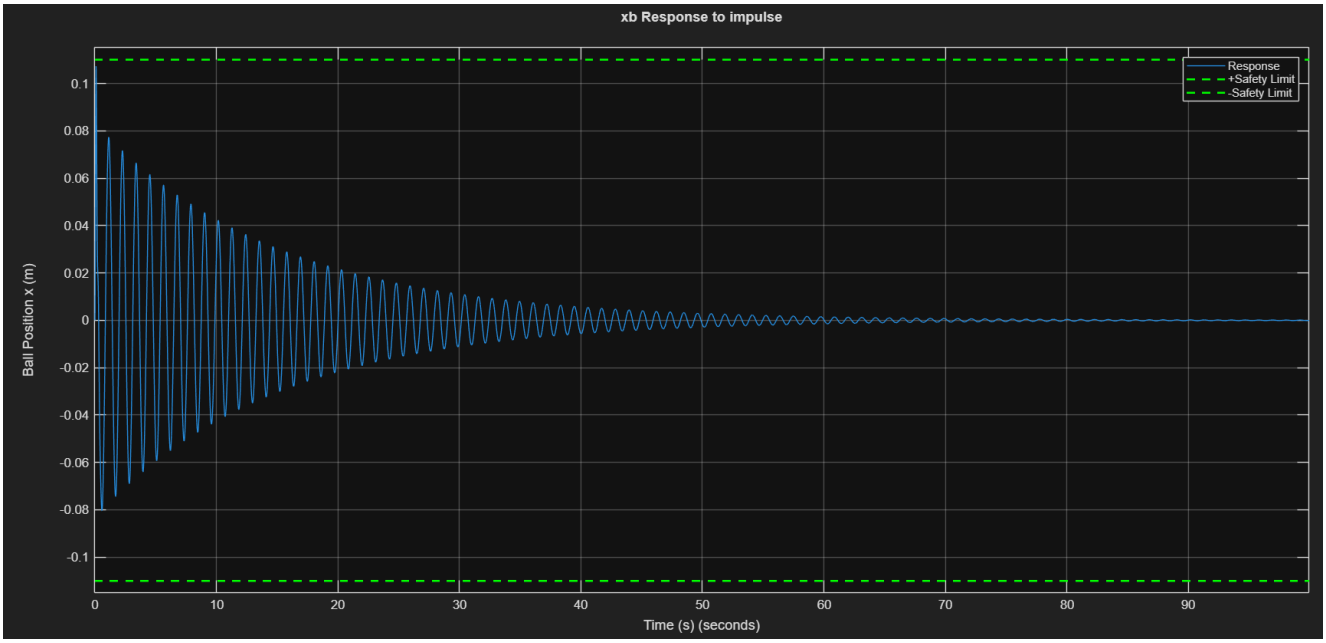


Figure 3.2: Impulse response of the Outer PID Loop

3.2.2 Outer Loop Design (Ball Position Control)

Using the closed inner loop, the effective plant becomes:

$$G_{x_b,\text{outer}}(s) = \frac{x(s)}{\theta_{\text{ref}}(s)}.$$

A second PID was tuned:

$$K_{px_b} = 7, \qquad K_{ix_b} = 0.01, \qquad K_{dx_b} = 5.$$

This controller generates the reference tilt angle θ_{ref} required to move the ball to the desired position x_{ref}.

3.3 LQR: Design Strategy and Justification

To achieve optimal stabilization of the underactuated system, a Linear Quadratic Regulator (LQR) was designed. Unlike pole placement, which requires the manual selection of closed-loop poles, LQR determines the optimal gain matrix K that minimizes a quadratic cost function J, balancing state error against control effort:

$$J = \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt \tag{3.2}$$

where Q is the state-weighting matrix and R is the control-weighting matrix.

3.3.1 Selection of Weighting Matrices

The weighting matrices were initially estimated using **Bryson's Rule** ($Q_{ii} = 1/x_{max}^2$) and then iteratively tuned to satisfy the strict physical constraints of the ball-balancing robot.

The final weighting matrices used to calculate the optimal gain K are:

$$Q = \text{diag}\left(\begin{bmatrix} 1 & 1 & 30 & 80 & 1500 & 100 \end{bmatrix}\right), \quad R = 1 \tag{3.3}$$

3.3.2 Assumptions and System Restrictions

The validity of the proposed controller is subject to the following physical constraints, which heavily influenced the pole selection:

1. **Geometric Constraints (Ball Position):** The ball position is restricted to the top surface length, $|x_b| < d/2$.
2. **Linearity Region (Small Angle Assumption):** The feedback gain K was derived using a linearized model where $\sin(\theta) \approx \theta$. The controller is therefore valid only for operating ranges within approximately $0^\circ \pm 15^\circ$. Large disturbances beyond this range typically introduce non-linearities that the linear controller cannot compensate for.
3. **Actuator Saturation:** Considering the DC motors have a limited supply voltage. Placing poles too far into the left-half plane would result in large gains in the K matrix, which would demand voltages exceeding V_{max} for even small error states. The selected poles represent a trade-off between performance and feasible control effort to avoid saturation.

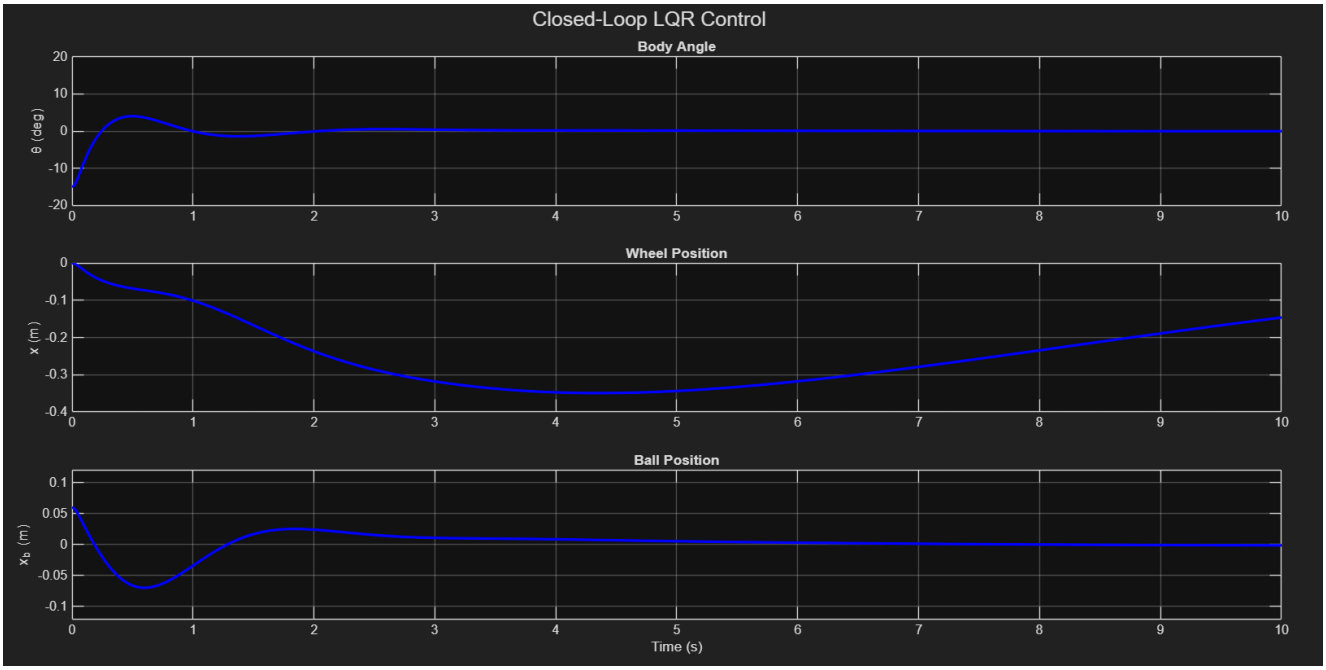


Figure 3.3: *Closed-loop LQR simulation showing the system response for an initial perturbation.*

3.4 Comparative Analysis: PID vs. LQR

PID Control (Cascaded SISO Design)

The PID controller was implemented using a two-loop nested structure:

1. an *inner* PID loop regulating the body angle θ,
2. an *outer* PID loop regulating the ball position x_b by commanding a desired angle θ_{ref}.

This cascaded structure is necessary because a single SISO PID acting on the plant cannot simultaneously regulate both angle and ball position. The inner loop was tuned to be fast and highly damped (large K_p and K_d), ensuring rapid stabilization of the dynamics of the inverted pendulum. The outer loop was tuned comparatively slower so that the ball motion did not excite the unstable inner subsystem.

Although this design works, it suffers from the classical drawbacks of SISO control applied to a coupled MIMO plant:

- tuning is sequential and non-systematic, changing outer loop gains affects the inner loop,
- cross-coupling between θ and x_b is not explicitly handled,
- performance strongly depends on the gain scheduling between loops.

In practice, the PID controller stabilized the robot, but the settling time for the ball position was significantly larger, and overshoot was present depending on the initial displacement.

LQR Control (Full-State Optimal Feedback)

The Linear Quadratic Regulator uses the full state vector

$$x = [\, x,\ \dot{x},\ \theta,\ \dot{\theta},\ x_b,\ \dot{x}_b \,]^T,$$

and computes a single control law $u = -Kx$ that simultaneously regulates all states based on an optimal cost function.

By appropriately selecting the state weighting matrix Q (with high penalties on x_b and $\dot{x}_b$), the LQR explicitly coordinates the chassis tilt and wheel motion to control the ball. Unlike the PID solution, no cascaded structure or special tuning heuristics are required, the coupling between states is handled automatically through the optimal gain matrix K.

Key advantages include:

- significantly smoother and faster settling of x_b,
- no need for multi-loop architecture,
- minimal overshoot,
- more predictable behavior under large initial disturbances.

3.4.1 Quantitative Performance Comparison: PID vs. LQR

This subsection presents a numerical comparison between the modern full-state LQR controller and the classical cascaded PID architecture. The following metrics were extracted from the time-domain responses of both controllers:

Table 3.1: Performance Metrics: LQR vs. PID

Metric	**LQR**	**PID (Inner)**	**PID (Outer)**
Body angle θ settling time	1.7 s	1.4 s	—
Body angle overshoot	33%	20%	—
Ball peak displacement x_b	0.07 m	—	0.107 m
Ball settling time x_b	2.6 s	—	55-60 s
Stability behaviour	Well-damped	Well-damped	Weakly damped
Control structure	SIMO	SISO	Cascaded SISO

Percentage Improvements

Ball Settling Time The most striking difference is in the ball-position dynamics. The LQR settles the ball position in approximately

$$T_{s,\mathrm{LQR}} \approx 1.7\,\mathrm{s},$$

while the cascaded PID outer loop takes

$$T_{s,\mathrm{PID}} \approx 55\text{-}60\,\mathrm{s}.$$

Body Angle Overshoot The body-angle overshoot reduces from 33% (LQR) to 20% (PID inner loop), corresponding to a

$$\frac{33-20}{33} \times 100 \approx 39\%$$

improvement. This is expected because the PID inner loop was tuned to be fast and aggressive.

Ball Peak Displacement The ball peak displacement for the PID case is $0.107\,\mathrm{m}$ as compared to that under LQR is $0.07\,\mathrm{m}$, a reduction of approximately

$$\frac{0.107-0.07}{0.107} \times 100 \approx 35.5\%.$$

Besides that, the PID-controlled response oscillates for almost a minute, whereas the LQR response damps out in under 2 seconds.

Interpretation

The LQR outperforms the PID architecture in all meaningful long-term stability metrics because it uses full-state feedback. By optimally coupling the dynamics of θ, $\dot{\theta}$, x_b, and $\dot{x}_b$, it enforces global stability and eliminates the slow, weakly-damped modes present in the PID outer loop.

In contrast, the PID controller acts only on local error signals. The inner loop regulates body angle efficiently, but the outer loop must indirectly control ball position by adjusting θ_{ref}, creating a slow, oscillatory, and weakly damped interaction. This leads to settling times that are two orders of magnitude worse than LQR.

Overall, LQR provides drastically superior stabilization of the underactuated ball dynamics and faster convergence of all states, making it the preferred controller for this system.

Chapter 4

Observer Design

4.1 Full State Observer

To implement advanced control strategies accurate knowledge of the system's internal states is required. While the Cart Position (x) and Chassis Angle (θ) are measured directly, the system lacks sensors for the linear velocity ($\dot{x}$), angular velocity ($\dot{\theta}$), and the ball's position and velocity ($x_b, \dot{x}_b$). To reconstruct these unmeasured states, a Full State Luenberger Observer was designed.

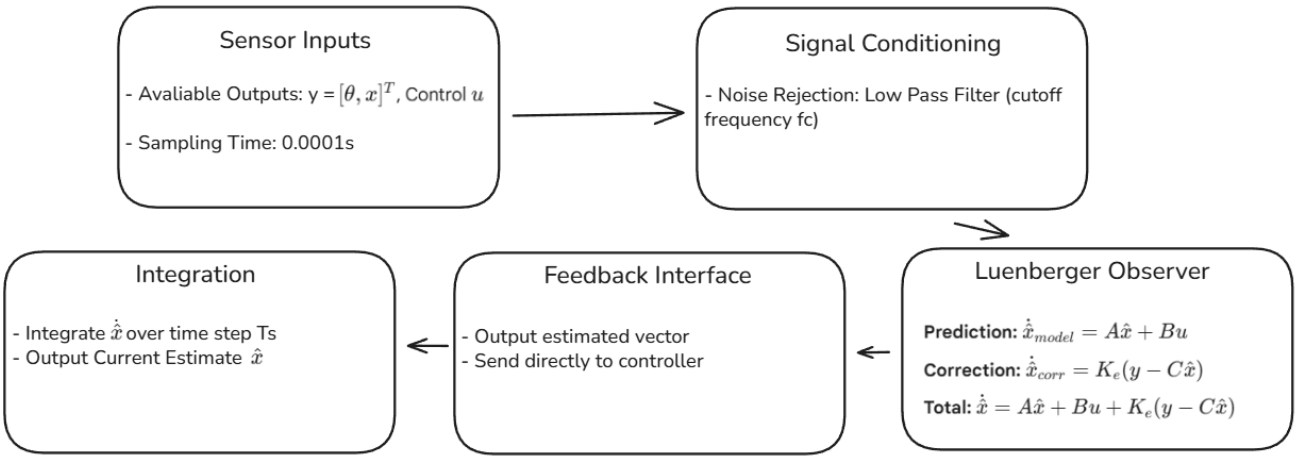


Figure 4.1: Architecture of the Full State Observer

4.1.1 Observer Theory and Dynamics

The observer is a computational model that runs in parallel with the physical plant. It uses the known control inputs $u(t)$ and the discrepancy between the measured outputs $y(t)$ and the estimated outputs $\hat{y}(t)$ to converge the estimated state $\hat{x}(t)$ to the true state $x(t)$.

The observer dynamics are governed by the equation:

$$\dot{\hat{x}} = A\hat{x} + Bu + Ke(y - C\hat{x}) \tag{4.1}$$

Here, the gain matrix Ke determines how heavily the observer relies on sensor data (y) versus its own internal model (A, B).

4.1.2 The Separation Principle

A key property utilized in this design is the Separation Principle, which states that the design of the Observer (Matrix Ke) and the Controller (Matrix K) can be performed independently. The poles of the observer were selected to be $4\times$ faster than the dominant poles of the controller. This ensures that the estimation error decays to zero rapidly, allowing the controller to act on accurate state information without significant lag.

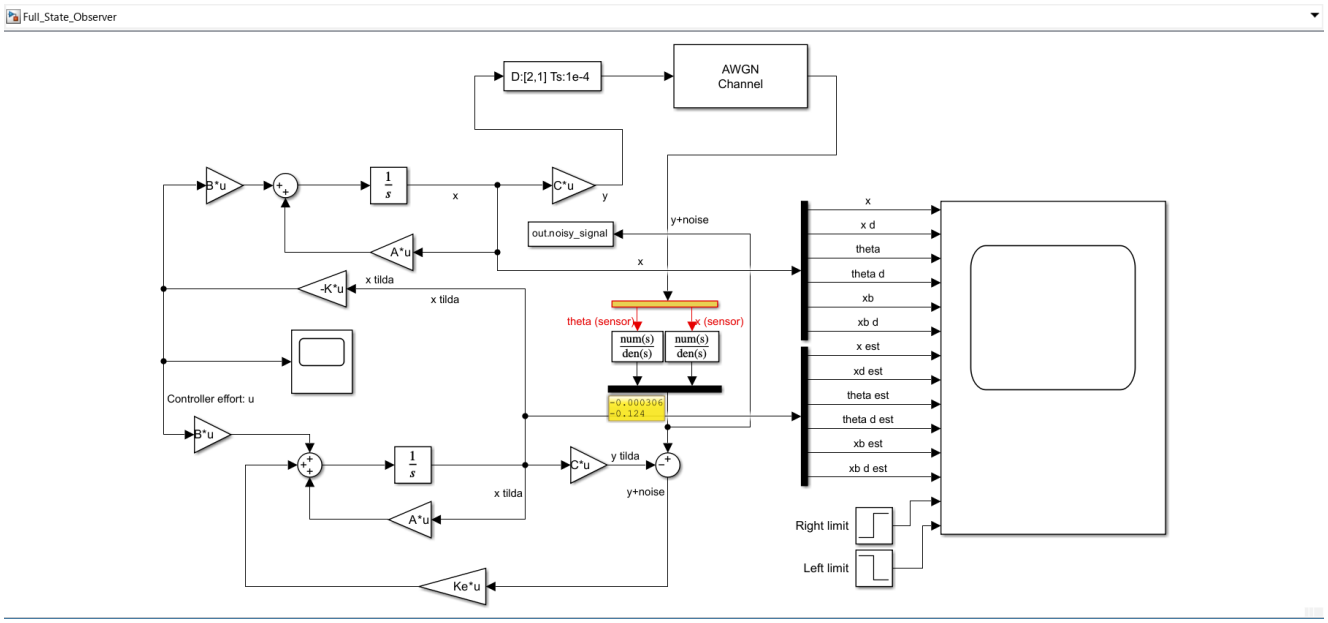


Figure 4.2: Full State Observer Block Diagram (Simulink)

4.2 Reduced Order Observer

While the Full State Observer estimates all system states, it is computationally inefficient because two of the states: Cart Position (x) and Chassis Angle (θ) are already measured with sensors. To improve efficiency and reduce computational overhead, a **Reduced Order Observer (ROO)** was designed to estimate only the four unmeasured states: linear velocity ($\dot{x}$), angular velocity ($\dot{\theta}$), ball position (x_b), and ball velocity ($\dot{x}_b$).

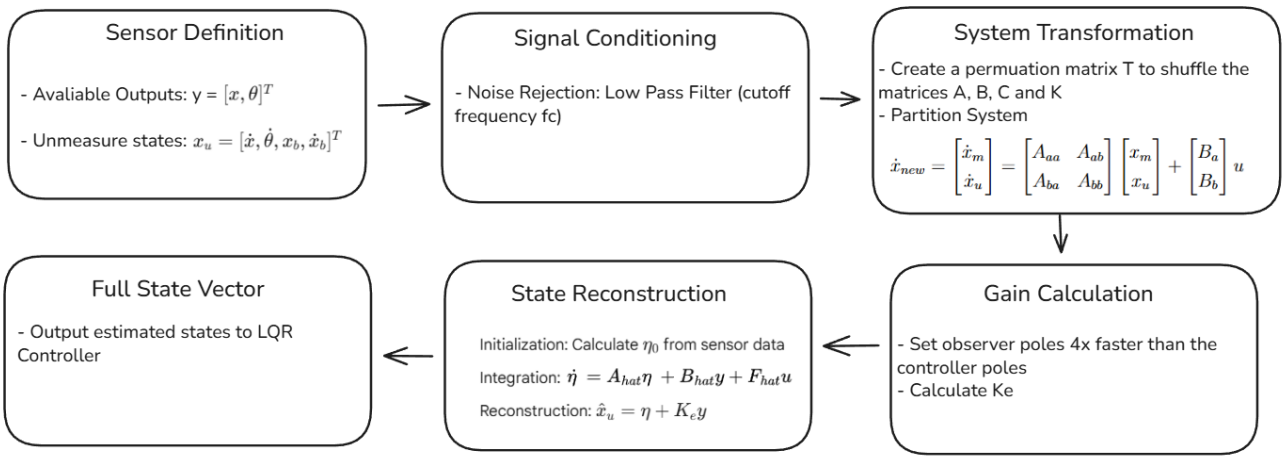


Figure 4.3: Flowchart detailing the Reduced Order Observer design process

4.2.1 System Partitioning and Transformation

The standard formulation for a reduced order observer requires the state vector to be partitioned into measured (x_m) and unmeasured (x_u) components. The original state vector was defined as $x = [x, \dot{x}, \theta, \dot{\theta}, x_b, \dot{x}_b]^T$.

To group the measured states, a similarity transformation matrix T was constructed to permute the state vector order to $x_{new} = [\theta, x, \dot{x}, \dot{\theta}, x_b, \dot{x}_b]^T$. This reordering enables the system matrices to be partitioned as follows:

$$\dot{x}_{new} = \begin{bmatrix} \dot{x}_m \\ \dot{x}_u \end{bmatrix} = \begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix} \begin{bmatrix} x_m \\ x_u \end{bmatrix} + \begin{bmatrix} B_a \\ B_b \end{bmatrix} u \tag{4.2}$$

where $x_m = [\theta, x]^T$ and $x_u = [\dot{x}, \dot{\theta}, x_b, \dot{x}_b]^T$.

4.2.2 Observer Dynamics

The dynamics of the reduced order observer are governed by the internal state vector $\eta \in \mathbb{R}^4$. The observer equations are derived to ensure the estimation error $e = x_u - \hat{x}_u$ converges to zero asymptotically:

$$\dot{\eta} = A_{hat}\eta + B_{hat}y + F_{hat}u \tag{4.3}$$

$$\hat{x}_u = \eta + K_e y \tag{4.4}$$

The observer matrices were computed in MATLAB as:

$$A_{hat} = A_{bb} - K_e A_{ab}$$
$$B_{hat} = A_{hat} K_e + A_{ba} - K_e A_{aa}$$
$$F_{hat} = B_b - K_e B_a$$

4.2.3 Gain Selection via Pole Placement

The observer gain matrix K_e was designed to ensure that the estimation error dynamics decay faster than the closed-loop system dynamics. To achieve this, the target poles for the observer were selected to be $4\times$ faster than the dominant eigenvalues of the LQR-controlled system $(A - BK)$.

$$p_{obs} \approx 4 \times \text{Re}(\text{eig}(A - BK_{LQR})) \tag{4.5}$$

Only the real parts of the poles were used to avoid introducing oscillatory dynamics into the estimator, ensuring a smooth convergence of the unmeasured states.

4.2.4 Initialization and Transient Suppression

A critical implementation detail for the ROO is the initialization of the internal state η. Unlike a full state observer where $\hat{x}(0)$ can simply be set to zero, the ROO state η is a linear combination of the unmeasured and measured states:

$$\eta(t) = x_u(t) - K_e y(t) \tag{4.6}$$

Initializing $\eta(0) = 0$ while the robot is tilted (e.g., $\theta_0 = 10^\circ$) would imply an erroneous initial condition for the unmeasured velocities, leading to a massive transient spike ("peaking") at startup. To prevent this, the initial condition for the integrator was explicitly calculated based on the sensor readings at $t = 0$:

$$\eta_0 = \hat{x}_{u,guess} - K_e y_{measured}(0) \tag{4.7}$$

where $\hat{x}_{u,guess}$ was set to zero (assuming the robot starts at rest). This calculation ensures the estimator begins in a consistent state, eliminating non-physical control spikes at startup.

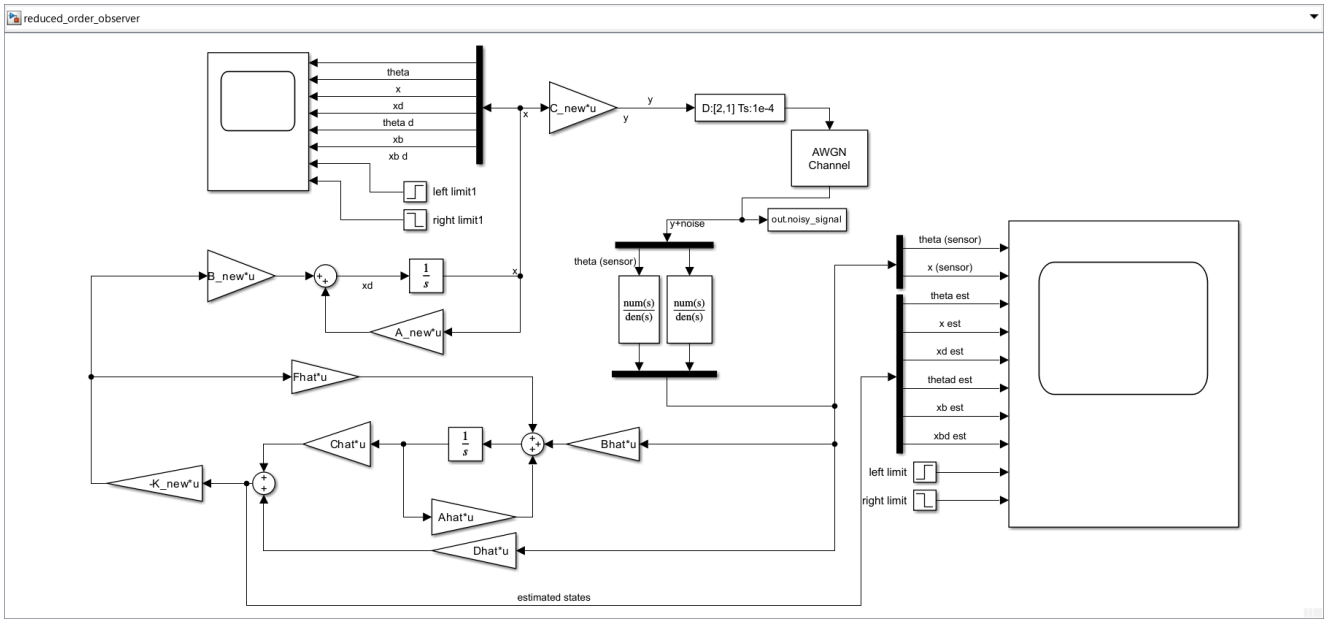


Figure 4.4: Reduced-Order Observer Block Diagram (Simulink)

4.3 Noise Analysis and Signal Filtering

To evaluate the robustness of the control system under realistic operating conditions, measurement noise was introduced into the simulation environment. This section details the noise characterization, the design of a signal conditioning filter, and the comparative performance of the observers under noisy conditions.

4.3.1 Noise Characterization

Simulink's *AWGN* block was inserted in line with the system output $y = [x, \theta]^T$. The noise parameters were selected to emulate good quality industrial sensors with SNR = 30dB.

An FFT analysis was performed on the raw sensor signal using MATLAB. As shown in Figure **??**, the noise exhibits a broadband spectrum with constant power density across the frequency range (1 to 5 kHz), confirming its "White Noise" characteristics.

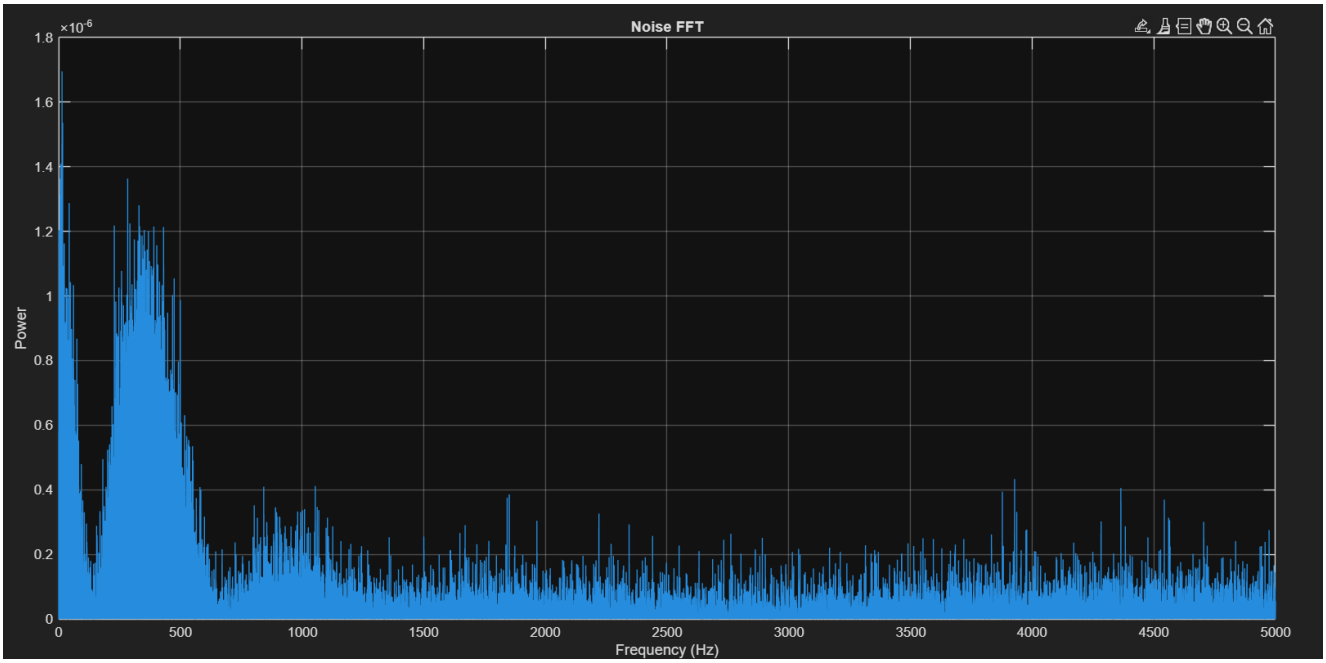


Figure 4.5: Unfiltered sensor noise for the Full State Observer

4.3.2 Filter Design

Since the observers differentiate the position signals to estimate velocities, high-frequency noise is naturally amplified. To mitigate this, a low-pass filter was designed.

- **Filter Type:** 1st Order Analog Low-Pass Filter (Transfer Function).
- **Cutoff Frequency:** $f_c = 15$ Hz ($\omega_c \approx 94$ rad/s).

Justification for Cutoff Frequency:
The cutoff frequency was selected based on the bandwidth separation principle. The dominant dynamics of the robot stabilization occur between 1-5 Hz. A cutoff of 15 Hz is sufficiently high to ensure the phase lag introduced by the filter is negligible, preventing instability in the closed-loop controller, while still effectively attenuating high-frequency sensor noise.

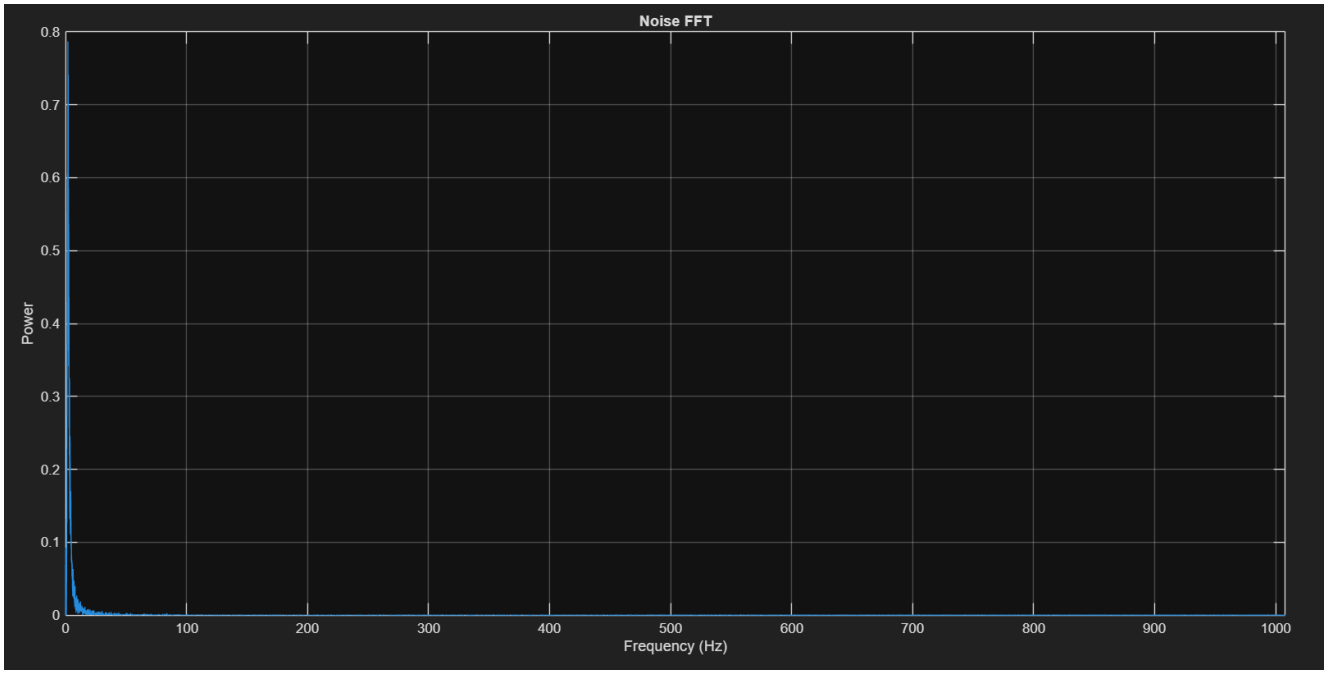


Figure 4.6: Filtered signal: Note the significant attenuation of noise power at frequencies above the 15 Hz cutoff.

4.3.3 Observer Performance Under Noise

The performance of both observers was evaluated using the noisy and filtered system output.

Reduced Order Observer (ROO)

The ROO reconstructs unmeasured states via the equation $\hat{x}_u = \eta + K_e y$. Because the measurement y feeds directly into the estimate, the ROO is highly sensitive to sensor noise.

- **Unfiltered:** The estimated velocities $(\dot{x}, \dot{\theta})$ exhibited violent high-frequency oscillations ("chatter").

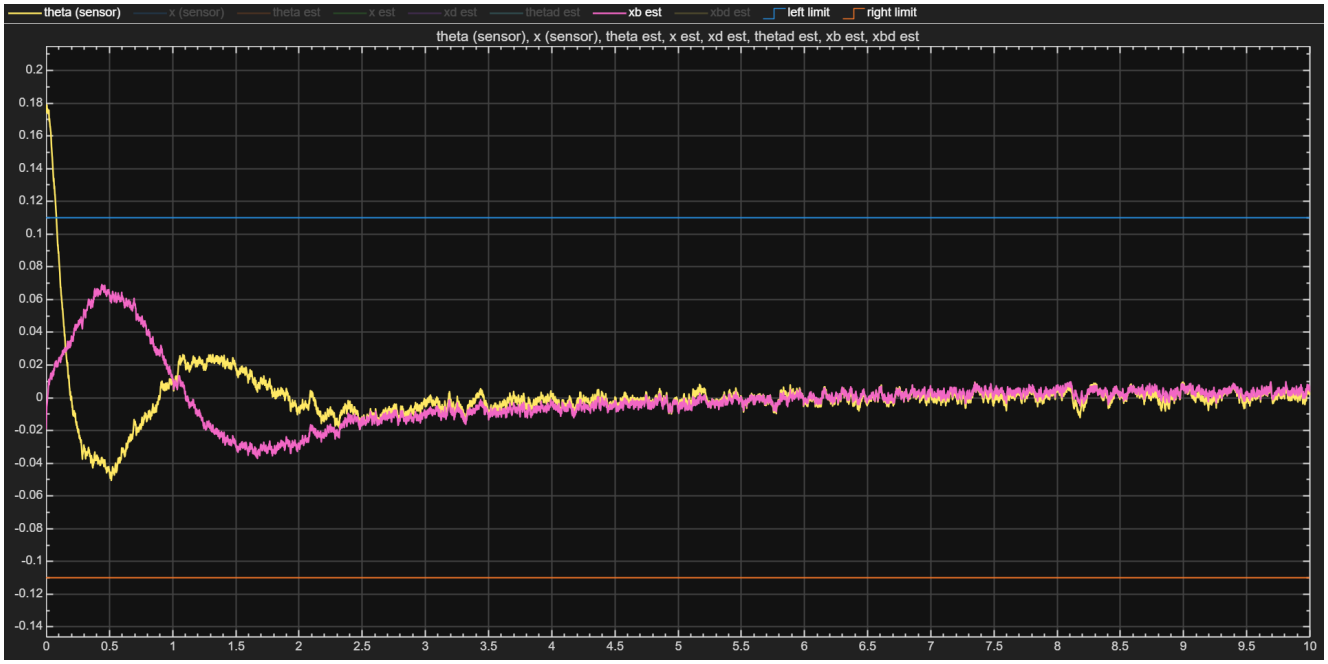


Figure 4.8: Performance of ROO under filtered noisy sensor data

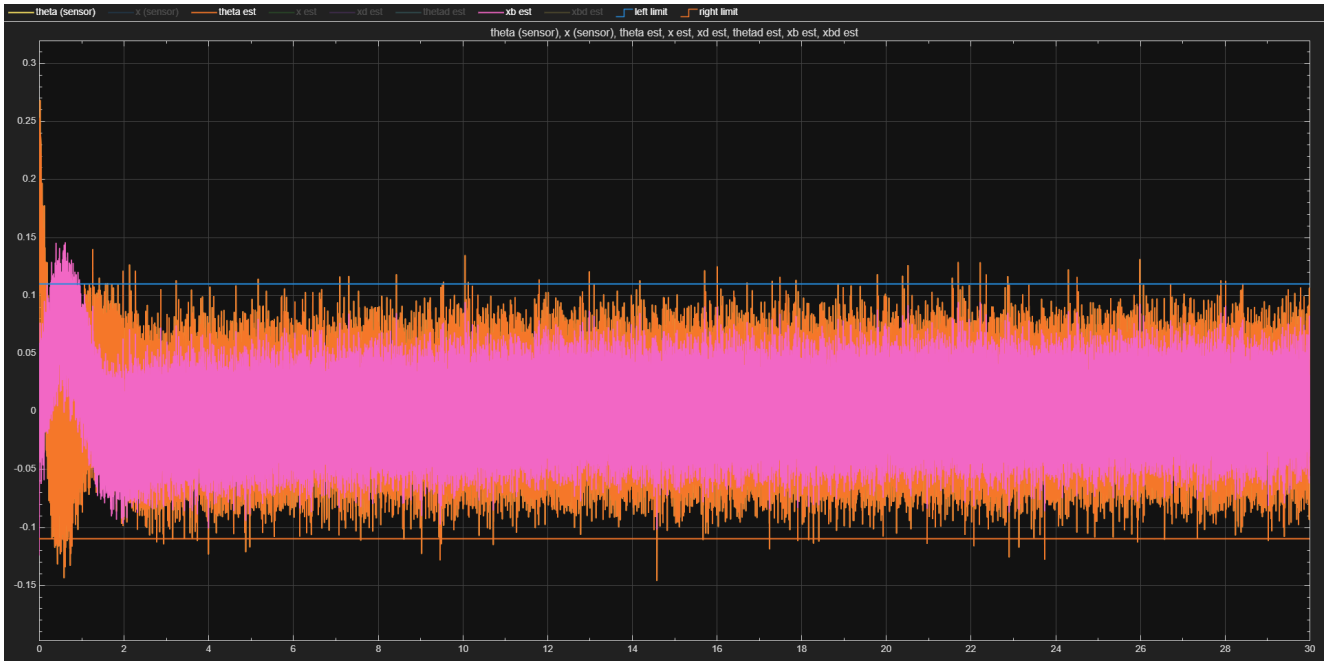


Figure 4.7: Performance of ROO under unfiltered noisy sensor data

- **Filtered:** With the 15 Hz pre-filter, the state estimates converged smoothly to the true values. As compared to the FSO, this one used much lesser controller effort.

Full State Observer (FSO)

The FSO exhibited faster observer dynamics under noisy measurements, which made its estimated states appear smoother than those of the ROO in the unfiltered case. However, this came at the cost of higher

sensitivity to noise in the control signal, since the aggressive observer gains amplified high-frequency components.

With the pre-filter applied, both the state estimates and controller effort improved significantly.

- **Unfiltered:** The FSO reacted faster to the noisy measurement compared to the ROO, resulting in a visually smoother estimate, but the control effort contained larger noise-amplified fluctuations.

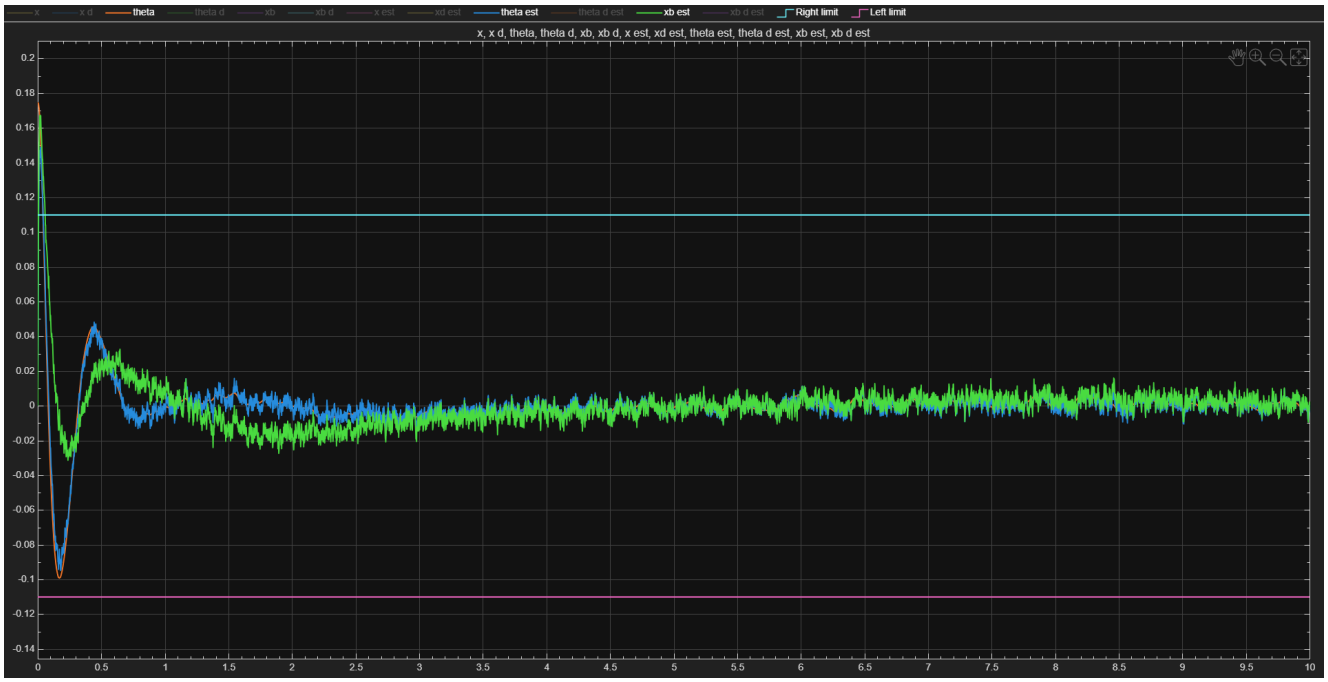


Figure 4.9: Performance of FSO under unfiltered noisy sensor data

- **Filtered:** With the 15 Hz pre-filter, the FSO performance improved, but the ROO still required less controller effort overall.

Figure 4.10: Performance of FSO under filtered noisy system input

Chapter 5

Conclusion

This project successfully encompassed the complete control system design lifecycle for a self-balancing robot with a a free-rolling ball. Moving from first-principles modeling to advanced state-space estimation, the work highlighted the distinct challenges posed by underactuated, open-loop unstable systems.

The system dynamics were derived using Lagrangian mechanics, revealing a highly coupled nonlinear system where the reaction torque from the wheel motors serves as the primary stabilizing mechanism for the chassis. This model was linearized around the unstable vertical equilibrium to facilitate the design of linear control strategies. Two distinct control architectures, PID and LQR, were designed and evaluated in simulation, supported by Full State and Reduced Order Observers to reconstruct unmeasured states from noisy sensor data.

While the PID controller provided a robust and intuitive solution for the primary stabilization task, the LQR controller coupled with a Reduced Order Observer is recommended as the final production solution. It offers the necessary multi-variable control authority to ensure the safety of the ball while minimizing computational load on the embedded processor.

References

- K. Ogata (2010). *Modern Control Engineering* Prentice Hall.
- YouTube Lectures: Brian Douglas, Christopher Lum, MIT OpenCourseWare