Modeling, Control, and State Estimation of a Ballbot System

SHREE HARI M G - ME23B201
ME4010 - Control Systems

Abstract

This report details the mathematical modeling, linearization, controller design, and state estimation for a ballbot system. A dynamic model is derived from first principles using Lagrangian mechanics, linearized about the upright equilibrium, and represented in state-space form. Two control strategies—Proportional-Integral-Derivative (PID) and Linear Quadratic Regulator (LQR)—are designed and compared. Furthermore, full-state and reduced-order observers are implemented to estimate unmeasured states in the presence of sensor noise. The system's robustness is analyzed using Fast Fourier Transform (FFT) analysis and signal filtering. Simulation results demonstrate that the LQR controller significantly outperforms the PID approach in settling time.

1 System Parameters and Modeling

1.1 Physical Parameters

The ballbot consists of a rigid body mounted on a single spherical wheel. The system parameters used for modeling are listed in Table 1. All values are converted to SI units for consistency.

Table 1: Physical Parameters of the Ballbot

Parameter	Symbol	Value
Gravity	g	$9.81\,\mathrm{m\,s^{-2}}$
Robot Mass	M_r	11.1 kg
Ball Mass	m_b	0.002 71 kg
Wheel mass	M_w	4.4 kg
Wheel Radius	R	0.328 m
Ball Radius	r	0.040 01 m
Body Height	d	0.198 05 m
Platform Height	h	0.328 05 m
Effective Lever Arm	$l = h + r$	0.368 06 m

The moments of inertia are calculated as:

$$I_b = \frac{2}{5} m_b r^2 \tag{1}$$

$$I_r = I_{r,com} + M_r d^2 \tag{2}$$

where $I_{r,com} = \frac{1}{12} M_r (h^2 + d^2)$ approximates the body as a rectangular prism.

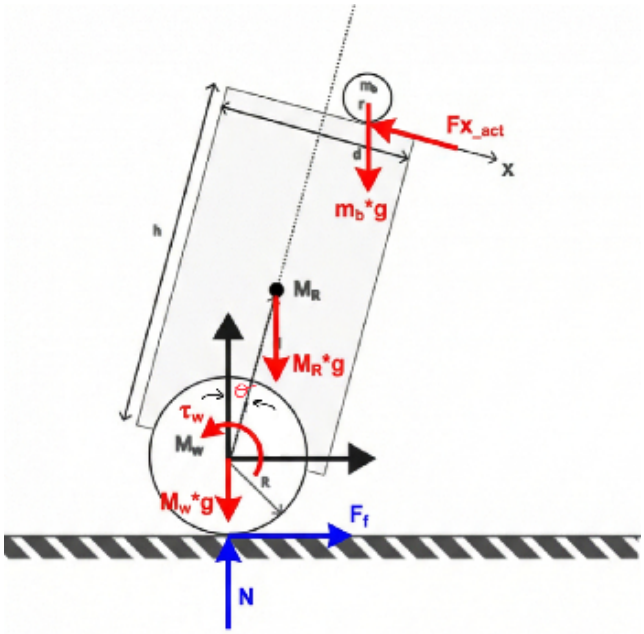

Figure 1: FBD

1.2 Lagrangian Dynamics

The generalized coordinates are defined as $q = [x,\ \theta]^T$, where x is the horizontal displacement of the ball and θ is the pitch angle of the body measured from the upright position. The Lagrangian formulation is used to derive the equations of motion. The total kinetic energy T contains

- translational and rotational kinetic energy of the ball,
- rotational kinetic energy of the body about the axle,
- a coupling term because motion of the body COM induces horizontal acceleration of the ball centre.

After expressing the ball centre velocity in terms of $\dot{x}$ and $\dot{\theta}$ (as in the SymPy notebook), the kinetic energy simplifies to

$$T = \tfrac{1}{2}\left(\tfrac{I_b}{r^2} + m_b\right)\dot{x}^2 + \tfrac{1}{2}(I_r + m_b l^2)\dot{\theta}^2 + m_b l\,\dot{x}\dot{\theta}. \tag{3}$$

The first term represents the effective translational inertia of the ball (its mass plus the rolling inertia I_b/r^2); the second term is the effective rotational inertia of the body and ball about the axle; the off–diagonal term

$m_b l\,\dot{x}\dot{\theta}$ captures the dynamic coupling between translation and pitch.

The potential energy U arises from gravity acting on the body COM at height d and the ball centre at height $h + r$:

$$U = g\big[M_r d\cos\theta + m_b\big((h+r)\cos\theta - x\sin\theta\big)\big]. \quad (4)$$

The term in $x\sin\theta$ couples horizontal displacement and tilt in the gravitational potential, which later produces the x–θ terms in the stiffness matrix.

With $L = T - U$, the Euler–Lagrange equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial \dot{q}_i}\Big) - \frac{\partial L}{\partial q_i} = Q_i$$

are applied for $q_1 = x$ and $q_2 = \theta$, where Q_i are the generalized forces due to the motor torque. The detailed symbolic derivation (performed in the attached SymPy notebook) yields two coupled, nonlinear second–order equations in x, θ and their derivatives.

1.3 Linearization and M_1, M_2, M_3 Matrices

To obtain a model suitable for linear control design, these equations are linearized about the upright equilibrium $x = 0,\ \theta = 0,\ \dot{x} = \dot{\theta} = 0$. Using the small–angle approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$, and neglecting products of small variables (terms such as $x\dot{\theta}^2$, $x\theta$ etc.), the dynamics can be written compactly as

$$M_1\ddot{q} = M_2 q + M_3 u, \qquad q = \begin{bmatrix} x \\ \theta \end{bmatrix}, \quad (5)$$

with

$$M_1 = \begin{bmatrix} \dfrac{I_b}{r^2} + m_b & m_b l \\ m_b l & I_r + m_b l^2 \end{bmatrix}, \quad (6)$$

$$M_2 = \begin{bmatrix} 0 & m_b g \\ m_b g & (M_r d + m_b l)g \end{bmatrix}, \quad (7)$$

$$M_3 = \begin{bmatrix} -m_b l \\ -(I_r + m_b l^2) \end{bmatrix}. \quad (8)$$

The matrix M_1 is the inertia matrix obtained directly from the quadratic form of the kinetic energy. Its diagonal entries are the effective translational and rotational inertias, while the symmetric off–diagonal element $m_b l$ represents inertial coupling between x and θ. The matrix M_2 collects the linearized gravity terms: there is no direct restoring force in x at the equilibrium ($M_2(1,1) = 0$), but gravity couples x and θ through the terms $\pm m_b g$, and provides an overall restoring torque on θ proportional to $M_r d + m_b l$. The input matrix M_3 encodes how the actuator torque appears in the generalized coordinates. Because the motor torque acts about the axle, it accelerates both the body inertia I_r and the ball mass at distance l, which is why $I_r + m_b l^2$ appears in the second entry and $m_b l$ in the first.

For state–space design, the second–order form is converted to first order by solving for $\ddot{q}$:

$$\ddot{q} = M_1^{-1}M_2 q + M_1^{-1}M_3 u = A_{\mathrm{sub}} q + B_{\mathrm{sub}} u.$$

Defining the state vector $X = [x,\ \dot{x},\ \theta,\ \dot{\theta}]^T$ gives the standard realization

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ A_{\mathrm{sub}}(1,1) & 0 & A_{\mathrm{sub}}(1,2) & 0 \\ 0 & 0 & 0 & 1 \\ A_{\mathrm{sub}}(2,1) & 0 & A_{\mathrm{sub}}(2,2) & 0 \end{bmatrix}, \quad (9)$$

$$B = -\begin{bmatrix} 0 \\ B_{\mathrm{sub}}(1) \\ 0 \\ B_{\mathrm{sub}}(2) \end{bmatrix}, \quad (10)$$

where $A_{\mathrm{sub}} = M_1^{-1}M_2$ and $B_{\mathrm{sub}} = M_1^{-1}M_3$ are computed numerically in the MATLAB script. The minus sign in B matches the sign convention used for the motor input in Simulink. The output matrix is chosen as $C = [0\ 0\ 1\ 0]$ so that only the tilt angle θ is measured.

2 Controller Design

2.1 PID Control Strategy

A cascaded PID control structure was implemented:

- **Inner Loop:** Stabilizes the tilt angle θ. Tuned for fast response ($K_p = 3000, K_i = 10, K_d = 25$).
- **Outer Loop:** Controls ball position x by manipulating the inner loop's setpoint. Tuned for slower position tracking ($K_p = 5, K_i = 0.01, K_d = 41$).

While stable, the PID response is sluggish due to the limited bandwidth of the outer loop required to maintain stability. (NOTE: C matrix for pid control system = [1 0 0 0; 0 0 1 0]. To measure both x and θ.

2.2 Linear Quadratic Regulator (LQR)

An LQR controller was designed to minimize the cost function $J = \int (X^T Q X + u^T R u)dt$. The weighting matrices were selected using Bryson's rule based on physical safety limits:

- $x_{max} = d/8$
- $\theta_{max} = 15^\circ$
- $\dot{x}_{max} = 0.5\,\mathrm{m\,s^{-1}}$
- $\dot{\theta}_{max} = 2.0\,\mathrm{rad\,s^{-1}}$
- $F_{max} = 5\,\mathrm{m\,s^{-2}}$

This yields $Q = \mathrm{diag}([1/x_{max}^2, \dots])$ and $R = 1/F_{max}^2$.

In the implemented design the state vector is $X = [x,\ \dot{x},\ \theta,\ \dot{\theta}]^T$ and the plant dynamics are

$$\dot{X} = AX + Bu, \qquad y = CX,$$

with A and B obtained from the linearized ballbot model as described in Section **??**. The LQR algorithm returns an optimal static state-feedback gain

$$K_{\mathrm{lqr}} = \begin{bmatrix} k_x & k_{\dot{x}} & k_\theta & k_{\dot{\theta}} \end{bmatrix},$$

and the control law is

$$u = -K_{\text{lqr}}\, X.$$

Substituting this into the plant yields the closed-loop matrix

$$A_{\text{cl}} = A - BK_{\text{lqr}} = A - B\begin{bmatrix} k_x & k_{\dot{x}} & k_\theta & k_{\dot{\theta}} \end{bmatrix},$$

which explicitly shifts all eigenvalues of A to the left-half plane. Structurally, the first column of A_{cl} is modified by $-k_x B$, the second by $-k_{\dot{x}} B$, and so on. This shows clearly how each feedback channel (x, $\dot{x}$, θ, $\dot{\theta}$) contributes a rank-one correction $-Bk_i$ to the open-loop dynamics.

Because the original pair (A, B) is controllable, any stable pole configuration for A_{cl} that is compatible with input limits can in principle be achieved by a suitable choice of K_{lqr}. In this work the desired pole locations are not specified directly; instead they emerge from the quadratic cost weights derived via Bryson's rule. Large entries in the diagonal of Q (for example on x and θ) force the corresponding columns of A_{cl} to move further left, resulting in faster decay of those states, whereas a larger R penalizes u and keeps the BK_{lqr} correction smaller, yielding a slower and less aggressive response. The final A_{cl} matrix therefore encodes a compromise between fast rejection of ball position and tilt deviations and the practical constraint on the maximum allowable wheel acceleration $F_{\max}$.

2.3 Comparison of PID and Full-State Feedback Control

The cascaded PID controller and the LQR full-state feedback controller act on the same physical system but exploit very different information patterns and control philosophies. The PID architecture is organized as an inner–outer loop structure: the inner PID loop regulates the body angle θ using u as the manipulated variable, while the outer loop adjusts the reference θ_{ref} to drive the ball position x to its setpoint. In contrast, the LQR controller computes the control input directly from the entire state vector, $u = -K_{\text{lqr}}X$, simultaneously shaping the dynamics of $x, \dot{x}, \theta$, and $\dot{\theta}$.

From a signal-flow perspective, the PID controller reacts locally to error signals. The inner loop uses the error $e_\theta = \theta_{\text{ref}} - \theta$ and its integral and derivative to stabilize the inverted-pendulum-like angle dynamics, largely independent of x. The outer loop then treats the closed inner loop as a new plant $G_x(s)$ and generates θ_{ref} from the position error $e_x = x_{\text{ref}} - x$. Because of the cascade, the outer loop must be significantly slower than the inner loop to avoid destabilizing interactions, which inherently limits the achievable bandwidth for x control.

The LQR scheme, by contrast, uses a single-loop structure that penalizes the entire state vector in the cost function. The gain vector $K_{\text{lqr}} = [k_x, k_{\dot{x}}, k_\theta, k_{\dot{\theta}}]$ is chosen such that the closed-loop poles are placed well inside the left-half plane while respecting actuator limits. Intuitively, k_θ and $k_{\dot{\theta}}$ strongly stabilize the tilt, while k_x and $k_{\dot{x}}$ shape the translational dynamics and damping. Because the controller has direct access to all states, it can coordinate corrections in x and θ in a coupled fashion, instead of separating them into nested loops.

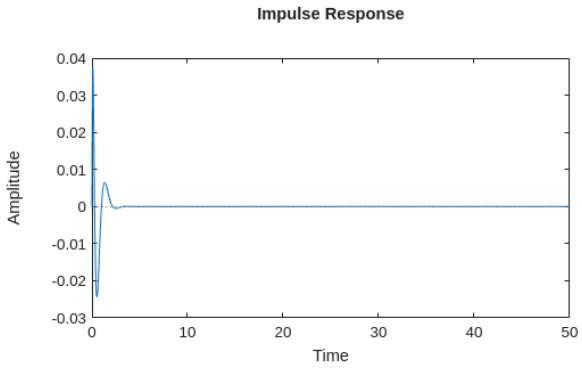

Figure 2: LQR: settling time vs x

Numerically, the difference in performance is stark. For a representative disturbance (impulse/initial-condition test with $x(0) = 5\,\text{mm}$ and $\theta(0) = 2\,\text{deg}$), the LQR controller achieves a ball position settling time of approximately 5 s, while the cascaded PID structure requires on the order of 300 s to reach a comparable tolerance band. This corresponds to an improvement factor of roughly $60\times$ in settling speed for x. Both controllers can be tuned to produce minimal steady-state error due to integral action (explicit in the PID and implicit via state feedback with appropriate reference handling in LQR), but the LQR consistently provides a faster, better-damped transient without violating the position and angle safety limits.

These differences impact the practical outcome in several ways. First, the LQR-controlled ballbot recovers from perturbations much more quickly, making it robust to real-world disturbances such as small pushes or modeling errors. Second, the cascaded PID must trade off outer-loop aggressiveness against inner-loop stability, leading to conservative gains and very slow position regulation. Finally, because LQR directly incorporates all state variables and actuator effort into a single optimization problem, it naturally balances tilt rejection and position regulation, whereas the PID design requires manual, iterative tuning of two separate loops and remains sensitive to plant parameter variations.

2.4 Performance Comparison

Both controllers were tested with an impulse disturbance. The LQR controller demonstrated superior performance:

- **LQR Settling Time:** $\approx 5\,\text{s}$
- **PID Settling Time:** $\approx 300\,\text{s}$

The LQR's full-state feedback allows it to aggressively stabilize both position and tilt simultaneously, whereas the cascaded PID is limited by the separation of time scales between the loops.

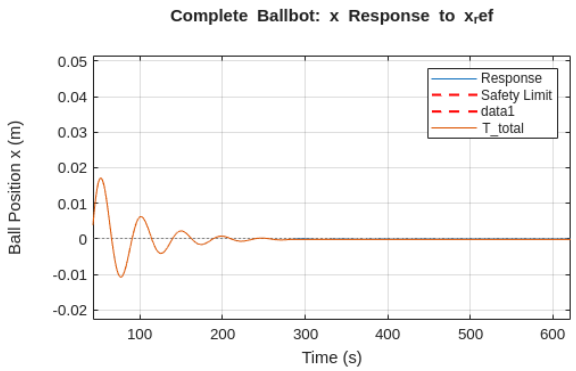

Figure 3: PID controller

3 State Estimation

In practice only the tilt angle θ is directly measured using an IMU, while the remaining states x, $\dot{x}$, and $\dot{\theta}$ must be reconstructed. This section describes the design of a full-state observer and a minimum-order observer .

3.1 Full-State Observer (LQE/Kalman Filter)

The linearized ballbot model can be written as

$$\dot{X} = AX + Bu, \qquad y = CX, \tag{11}$$

with $X = [x, \dot{x}, \theta, \dot{\theta}]^T$ and $y = \theta$. A continuous-time Luenberger/Kalman observer has the form

$$\dot{\hat{X}} = A\hat{X} + Bu + K_{\text{kalman}}\big(y - C\hat{X}\big), \tag{12}$$

where K_{kalman} is chosen such that $A - K_{\text{kalman}}C$ is stable and sufficiently faster than the closed-loop dynamics.

The gain K_{kalman} is obtained via the linear quadratic estimator (LQE) by specifying process-noise and measurement-noise covariances:

$$Q_{\text{est}} = \text{diag}\big(x^2_{\text{max,err}},\, \dot{x}^2_{\text{max,err}},\, \theta^2_{\text{max,err}},\, \dot{\theta}^2_{\text{max,err}}\big), \tag{13}$$

$$R_{\text{est}} = \sigma^2_\theta, \tag{14}$$

where the diagonal entries reflect expected model errors (e.g. 2 mm in position, 0.3° in angle) and σ_θ corresponds to the IMU angle noise standard deviation (e.g. 0.1°). With these choices, all observer poles of $A - K_{\text{kalman}}C$ lie well to the left of the LQR closed-loop poles, so that the estimate $\hat{X}$ converges rapidly.

In the controller implementation, the true state in the LQR law

$$u = -K_{\text{lqr}}X \tag{15}$$

is replaced by the observer estimate,

$$u = -K_{\text{lqr}}\hat{X}, \tag{16}$$

yielding a full-state-feedback structure based entirely on measured θ and the model-based reconstruction of $(x, \dot{x}, \dot{\theta})$.

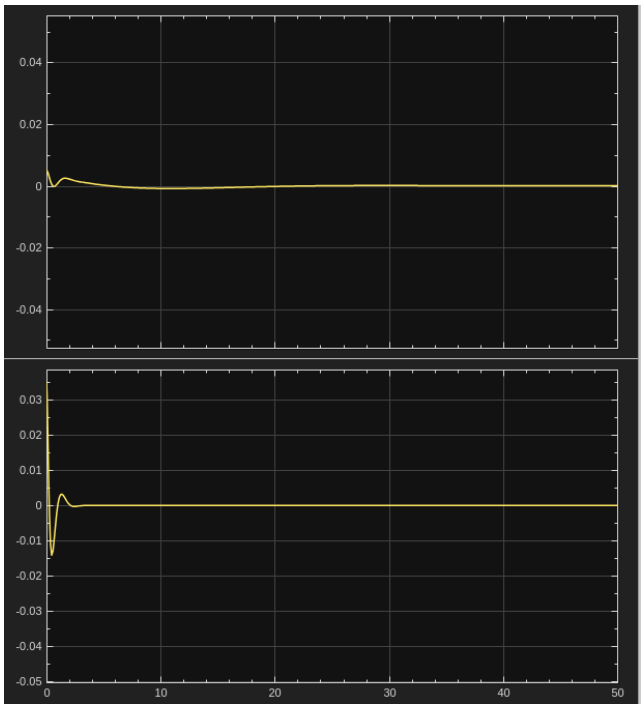

Figure 4: Full order observer, top:x, bottom:θ

3.2 Minimum-Order Observer and η-Transformation

Since only θ is measured, a reduced-order observer can be constructed that estimates only the unmeasured states. Partition the state vector and system matrices as

$$X = \begin{bmatrix} x_u \\ x_a \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \dot{\theta} \\ \theta \end{bmatrix}, \quad A = \begin{bmatrix} A_{bb} & A_{ba} \\ A_{ab} & A_{aa} \end{bmatrix}, \quad B = \begin{bmatrix} B_b \\ B_a \end{bmatrix}, \tag{17}$$

where $x_a = \theta$ (measured, index set $i_a = [3]$) and $x_u = [x, \dot{x}, \dot{\theta}]^T$ (unmeasured, index set $i_b = [1, 2, 4]$). The output equation becomes

$$y = C_b x_u + C_a x_a, \tag{18}$$

with C_b and C_a extracted from the original C.

The minimum-order observer is based on the transformation

$$\eta = x_u - L_{\min} x_a, \tag{19}$$

where $L_{\min}$ is chosen such that the dynamics of η are stable and of dimension equal to the unmeasured part. Differentiating and substituting the plant equations,

$$\dot{x}_u = A_{bb}x_u + A_{ba}x_a + B_b u, \tag{20}$$

$$\dot{x}_a = A_{ab}x_u + A_{aa}x_a + B_a u, \tag{21}$$

gives

$$\begin{aligned} \dot{\eta} = {} & (A_{bb} - L_{\min}A_{ab})\, x_u \\ & + (A_{ba} - L_{\min}A_{aa})\, x_a \\ & + (B_b - L_{\min}B_a)\, u. \end{aligned} \tag{22}$$

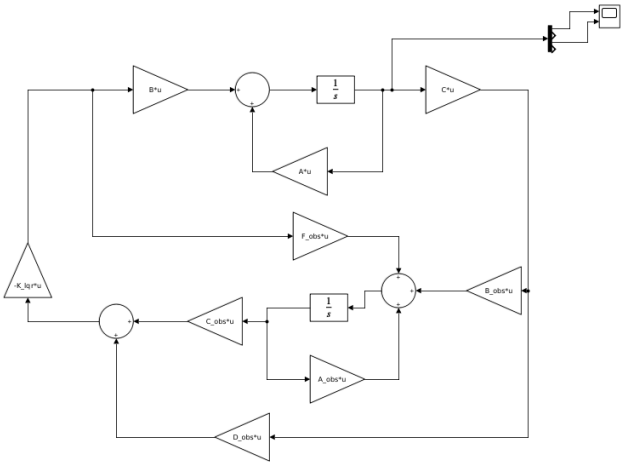

Figure 5: Minimum order observer

Replacing $x_u = \eta + L_{\min} x_a$ leads to the reduced-order observer dynamics

$$\dot{\hat{\eta}} = A_{\text{obs}}\hat{\eta} + B_{\text{obs}} x_a + F_{\text{obs}} u, \tag{23}$$

with

$$A_{\text{obs}} = A_{bb} - L_{\min} A_{ab}, \tag{24}$$
$$B_{\text{obs}} = A_{ba} - L_{\min} A_{aa} + A_{\text{obs}} L_{\min}, \tag{25}$$
$$F_{\text{obs}} = B_b - L_{\min} B_a. \tag{26}$$

The unmeasured state estimate is reconstructed as

$$\hat{x}_u = \hat{\eta} + L_{\min} x_a, \tag{27}$$

and the full estimated state used for feedback is

$$\hat{X} = \begin{bmatrix} \hat{x}_u \\ x_a \end{bmatrix}. \tag{28}$$

In the block diagram of Fig. 5, the matrices $\hat{A}$, $\hat{B}$, and $\hat{F}$ correspond to A_{obs}, F_{obs}, and B_{obs}, while the outer transformation block implements the mapping from $\hat{\eta}$ and y to $\hat{X}$ via suitable C_{obs} and D_{obs}.

The gain $L_{\min}$ is designed with an LQE applied to the reduced subsystem (A_{bb}, A_{ab}) using process-noise and measurement-noise covariances $Q_{\min}$ and $R_{\min}$. These are tuned such that the reduced-order observer converges at a rate comparable to the full Kalman filter while avoiding excessive overshoot in the reconstructed x and $\dot{x}$.

3.3 Noise, FFT Analysis, and Filtering

To emulate realistic sensor behavior, Simulink's noise block is inserted in series with the measured angle output, producing a noisy measurement $y_{\text{noise}}(t)$. The resulting time series is exported to the workspace and analyzed in the frequency domain using the Fast Fourier Transform (FFT). The power spectral density reveals that most of the measurement noise is concentrated at frequencies well above the dominant closed-loop bandwidth.

Based on this analysis, a second-order Butterworth low-pass filter with cutoff frequency $f_c \approx 20\,\text{Hz}$ is designed and inserted between y_{noise} and the observers.

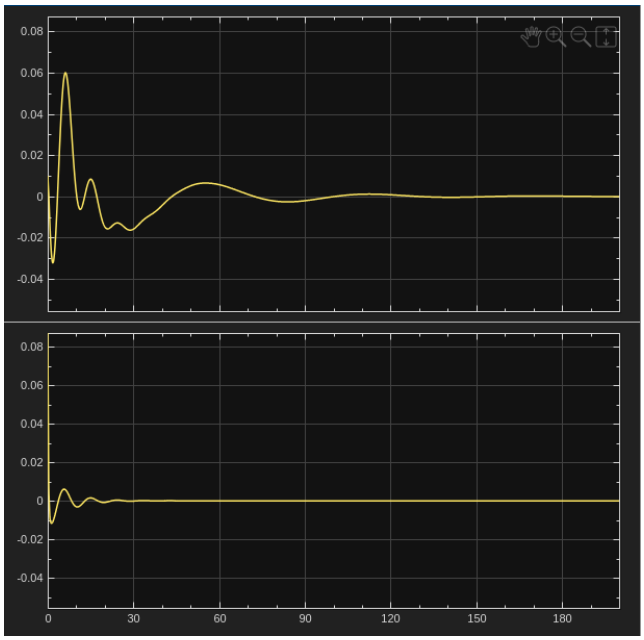

Figure 6: Minimum order observer, top:x, bottom:θ

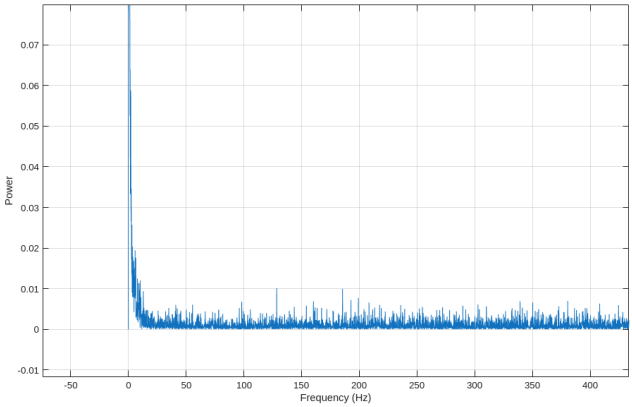

Figure 7: Noise

This filter preserves the relevant ballbot dynamics while attenuating high-frequency noise. Both the full-state and reduced-order observers are then driven by the filtered signal, and their performance is evaluated in simulation in terms of convergence speed, estimation error magnitude, and the smoothness of the resulting control input.

4 Noise Analysis and Filtering

4.1 Noise Characterization (FFT)

Realistic measurement noise was injected into the simulation using Simulink's white-noise block at the output. A Fast Fourier Transform (FFT) of the noisy θ signal confirmed a broadband noise spectrum.

4.2 Filter Design

The noisy output signal was logged from Simulink and exported to MATLAB for spectral analysis. Specifically, the noise time series $y_{\text{noise}}(t)$ and its time vec-

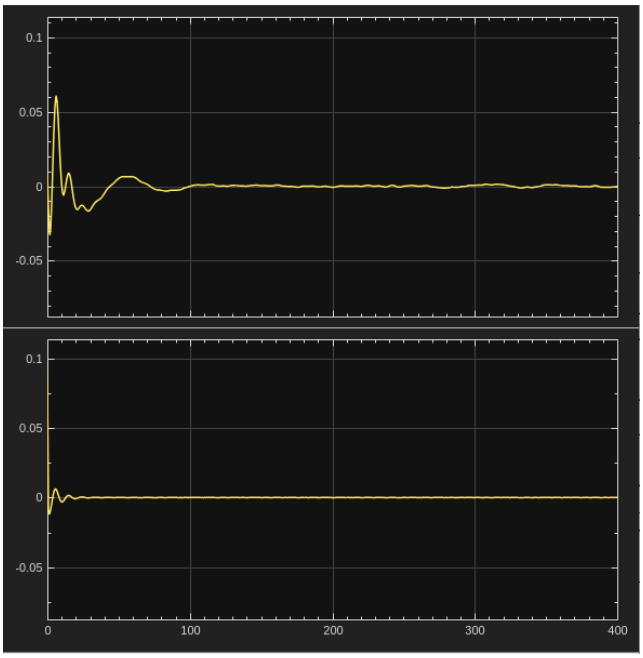

Figure 8: Observer with noise. Top:x and bottom:θ

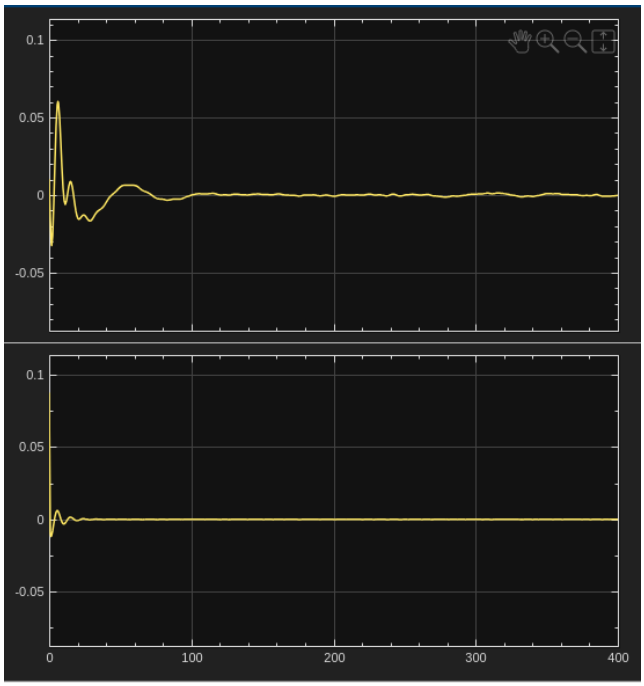

Figure 9: Min order observer with noise. Top:x and bottom:θ

tor were saved from the simulation (using commands such as `y_noise = out.y_noise.signals.values;` and `ti = out.y_noise.time;`) and processed offline. A Fast Fourier Transform (FFT) of $y_{\text{noise}}(t)$ was then computed in MATLAB, and the resulting power spectrum was inspected to identify the dominant noise frequency range. Based on this spectrum, a cutoff frequency of $f_c \approx 20\,\text{Hz}$ was selected for the low-pass Butterworth filter, so that the filter attenuates the high-frequency noise components while preserving the relevant ballbot dynamics.‘‘‘

5 Results and Discussion

5.1 Observer Performance with Noise

Both observers were tested under noisy conditions. The convergence times (settling to within 2% error) were recorded:

Table 2: Observer Convergence Times with Noise

Observer Type	Convergence Time
Full-Order Observer	$\approx 90\,\text{s}$
Reduced-Order Observer	$\approx 120\,\text{s}$

5.2 Analysis

To evaluate the observers in closed loop, the true plant state X was used only for logging, while the LQR input was computed from the estimates, $u = -K_{\text{lqr}}\hat{X}$. With the full-state observer, the resulting trajectories for $x(t)$ and $\theta(t)$ were practically identical to the ideal full-state-feedback response, showing that the estimated state is accurate enough to reproduce the desired behaviour when used directly in feedback. The estimation error $X - \hat{X}$ decayed rapidly to a small noise band, confirming that the poles of $A - K_{\text{kalman}}C$ are sufficiently fast.

A reduced-order observer was then designed so that only the unmeasured states $(x, \dot{x}, \dot{\theta})$ are estimated and the measured angle θ is passed through. Its gain L_{min} was tuned so that, under the same LQR law $u = -K_{\text{lqr}}\hat{X}$, the ball position and tilt remained within the same safety limits as with the full observer. Although the reduced-order estimator converged more slowly, it still met the qualitative performance requirement that the closed loop with estimated state behaves like the full-state-feedback system.

To test robustness to sensing errors, Simulink white noise was injected at the output channel and both observers were driven by this noisy measurement. In this setting the full-order observer converged to its steady error band in roughly 90 s, while the reduced-order observer required about 120 s, indicating greater sensitivity to the tuning of Q_{min} and R_{min}. The noisy output was then logged and analysed using an FFT; the resulting spectrum showed that most of the noise energy lay at frequencies higher than the dominant closed-loop dynamics. On this basis a second-order Butterworth low-pass filter with cutoff $f_c \approx 20\,\text{Hz}$ was introduced in series with the measurement. With the filtered signal both observers produced noticeably smoother state estimates, and the LQR control input exhibited reduced high-frequency chatter, while the overall closed-loop behaviour and safety constraints were preserved.

6 Conclusion

This project took the ballbot from first-principles modelling all the way to robust, observer-based feedback control. Starting from a Lagrangian formulation, the

nonlinear equations of motion were derived and linearized about the upright equilibrium, yielding a compact state-space model that is both controllable and observable from a single tilt measurement. On top of this model, two control strategies were implemented: a cascaded PID design based on transfer functions and root-locus intuition, and an optimal LQR controller based on state-space methods. The LQR controller, tuned via Bryson's rule, reshaped the closed-loop matrix to achieve much faster and better-damped transients than the PID design while still respecting position, angle, and actuator limits. Full-state and reduced-order observers were then added so that the same LQR law could be driven by estimated states only. Under simulated sensor noise, both observers maintained stability and constraint satisfaction, with the full-order estimator converging more quickly and the reduced-order version offering a lighter implementation at the cost of slower convergence. FFT-based analysis of the noisy output motivated the use of a simple Butterworth low-pass filter, which significantly reduced estimation variance and actuator chatter without degrading dynamics. Overall, the work demonstrates how a consistent modelling framework, combined with optimal state-feedback and appropriately tuned observers, leads to a ballbot controller that is fast, robust to noise, and systematically justifiable rather than purely heuristic.

References

[1] N. S. Nise, *Control Systems Engineering*, 7th ed., Wiley, 2015.

[2] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Hemisphere Publishing, 1975.

[3] R. Tedrake, "Chapter 8: Linear Quadratic Regulators," in *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*, MIT OpenCourseWare, accessed 2025. (Used as a reference for Bryson's rule–based LQR tuning.)