Modelling of a Two-wheeled Self-Balancing Robot

Shwetha V ME23B073

Nov 2025

1 System Modelling

1.1 Physical System Description

The system considered is a two–dimensional self–balancing robot consisting of a single wheel of mass M_w and radius R, supporting a rigid body of mass M_R whose centre of mass (COM) lies at a distance l from the wheel axle. A schematic is shown in Fig. 1. The wheel rolls without slip on flat ground, producing a kinematic relation between the translational motion of the robot and the rotation of the wheel. The configuration of the system is

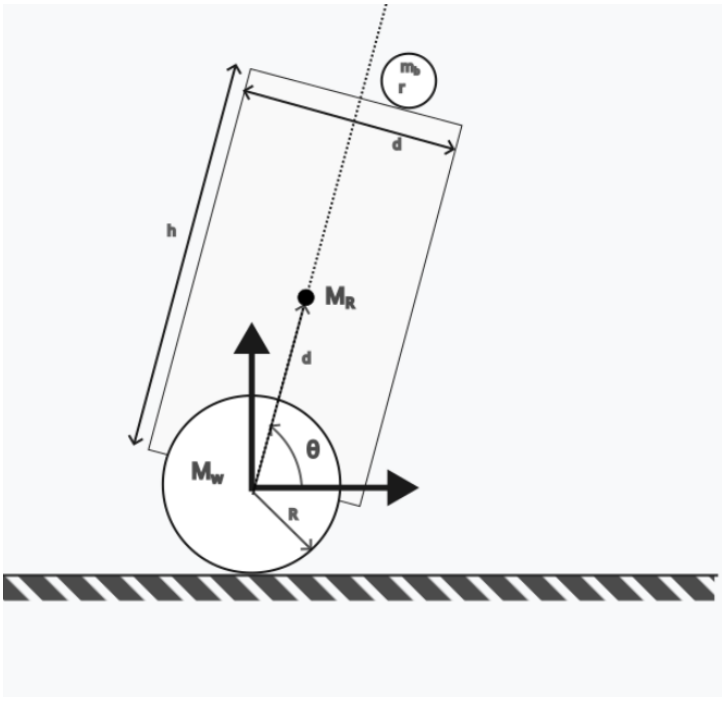

Figure 1: Schematic

fully described by two generalized coordinates:

- θ: body pitch angle measured about the wheel axle (upright position $\theta = 0$),
- ϕ: rotation angle of the wheel.

An input torque u is applied at the wheel, causing both wheel acceleration and body rotation through dynamic coupling. The system behaves as a classical inverted pendulum mounted on a reaction wheel, but with significantly stronger coupling due to the finite wheel mass and COM height.

1.2 Modelling Assumptions

- Motion is restricted to the 2D plane.
- No-slip rolling condition: $\dot{x} = R\dot{\phi}$.
- The robot body is a rigid rod of height h with centre of mass located at distance l above the axle.
- Viscous damping b_θ (body) and b_ϕ (wheel) model frictional losses.
- The wheel inertia and body inertia are modelled as

$$J_w = \frac{1}{2} M_w R^2, \tag{1}$$

$$J_b = \frac{1}{12} M_R h^2 + M_R (h/2 - l)^2, \tag{2}$$

  using parallel–axis theorem.

1.3 System equations

The equations are derived via Newton–Euler–Lagrange formalism. The body experiences gravitational torque and reaction torques transmitted through the wheel. The generalized coordinates are $q = [\theta, \phi]^T$.

The total kinetic energy is

$$T = \frac{1}{2}(J_b + M_R l^2)\dot{\theta}^2 + M_R l R \dot{\theta}\dot{\phi} + \frac{1}{2}\left(J_w + (M_R + M_w)R^2\right)\dot{\phi}^2, \tag{3}$$

and the potential energy is

$$V = M_R g l \cos\theta. \tag{4}$$

The equations of motion follow from the Euler–Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = Q, \quad \mathcal{L} = T - V,$$

where $Q = [-b_\theta \dot{\theta},\ u - b_\phi \dot{\phi}]^T$ incorporates dissipative and actuation torques.

After algebraic simplification, the coupled nonlinear dynamics become:

$$A\,\ddot{\theta} + B\,\ddot{\phi} = M_R g l \sin\theta - b_\theta \dot{\theta}, \tag{5}$$

$$B\,\ddot{\theta} + C\,\ddot{\phi} = u - b_\phi \dot{\phi}, \tag{6}$$

where

$$A = J_b + M_R l^2, \tag{7}$$

$$B = M_R l R, \tag{8}$$

$$C = J_w + (M_R + M_w) R^2. \tag{9}$$

Solving (5)–(6) for $\ddot{\theta}$ and $\ddot{\phi}$ gives

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}^{-1} \begin{bmatrix} M_R g l \sin\theta - b_\theta \dot{\theta} \\ u - b_\phi \dot{\phi} \end{bmatrix}. \tag{10}$$

1.4 State–Space Representation

Define the state vector

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}.$$

Then the dynamics take the form

$$\dot{x}_1 = x_2, \tag{11}$$

$$\dot{x}_3 = x_4, \tag{12}$$

$$\dot{x}_2 = f(x_1, x_2, x_4, u), \tag{13}$$

$$\dot{x}_4 = g(x_1, x_2, x_4, u), \tag{14}$$

where f, g follow directly from (10). Thus the nonlinear state–space system is:

$$\dot{x} = \begin{bmatrix} x_2 \\ f(x,u) \\ x_4 \\ g(x,u) \end{bmatrix}.$$

1.5 Equilibrium (Fixed) Points

At equilibrium, $\dot{x} = 0$. Thus:

$$\dot{\theta} = \dot{\phi} = \ddot{\theta} = \ddot{\phi} = 0.$$

From (5)–(6):

$$M_R g l \sin\theta^* = 0, \tag{15}$$

$$u^* = 0. \tag{16}$$

Solutions:

$\theta^* = 0$ (upright, unstable), $\theta^* = \pi$ (hanging down, stable).

For the balancing problem, we linearize about the upright equilibrium:

$$x^* = \begin{bmatrix} 0 & 0 & \phi_0 & 0 \end{bmatrix}^T, \qquad u^* = 0.$$

1.6 Linearization About the Upright Position

Using first–order Taylor expansion:

$$\dot{x} = f(x,u) \approx f(x^*,0) + \left.\frac{\partial f}{\partial x}\right|_{x^*}(x - x^*) + \left.\frac{\partial f}{\partial u}\right|_{x^*}(u - 0).$$

Let the linearized system be

$$\dot{x} = Ax + Bu.$$

The Jacobian matrices evaluated at ($\theta = 0$, $\dot{\theta} = 0$, $\dot{\phi} = 0$) are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M_R g l\, C}{AC - B^2} & -\frac{b_\theta C}{AC - B^2} & 0 & \frac{B\, b_\phi}{AC - B^2} \\ 0 & 0 & 0 & 1 \\ -\frac{M_R g l\, B}{AC - B^2} & \frac{b_\theta B}{AC - B^2} & 0 & -\frac{b_\phi A}{AC - B^2} \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ \frac{C}{AC - B^2} \\ 0 \\ -\frac{B}{AC - B^2} \end{bmatrix}.$$

This linear model will be used for PID controller design, state feedback, LQR, and observer design in later sections.

1.7 Model Validation

The nonlinear equations derived for the self-balancing robot are validated through MATLAB simulations. Validation was performed using (i) a 2-D visual animation of the robot, and (ii) numerical simulation of the nonlinear ODEs under different torque inputs.

A MATLAB animation is created using the simulated states $\theta(t)$ and $\phi(t)$ to depict the wheel, body and balancing ball. A representative frame is shown in Fig. 2. The animation confirms that a small initial tilt $\theta(0) = 0.02$ rad causes the body to fall and the wheel to rotate accordingly, consistent with inverted-pendulum behaviour.

Table 1: Given and calculated physical parameters of the self-balancing robot

Parameter	**Symbol**	**Value**
Body mass	M_R	9.14 kg
Wheel mass	M_w	2.10 kg
Wheel radius	R	0.295 m
Body height	h	0.35004 m
COM distance from pivot	l	0.22504 m
Ball radius	r_b	0.04001 m
Body inertia	J_b	0.11620 kg m^2
Wheel inertia	J_w	0.09138 kg m^2
Damping (body)	b_θ	0.02 N m s
Damping (wheel)	b_ϕ	0.02 N m s
Gravity	g	9.81 m/s^2

Table 2: Derived inertia and coupling constants

Quantity	**Expression**	**Value**
A	$J_b + M_R l^2$	0.5801 kg m^2
B	$M_R l R$	0.6070 kg m^2
C	$J_w + (M_R + M_w)R^2$	1.0695 kg m^2

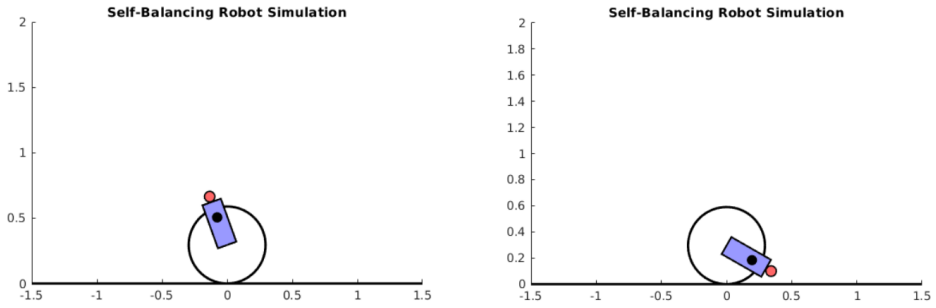

Figure 2: 2-D visual simulation of the robot motion.

2 Controller Design

2.1 Transfer Functions of the Linearized System

With the linearized state–space model

$$\dot{x} = Ax + Bu, \qquad y = Cx,$$

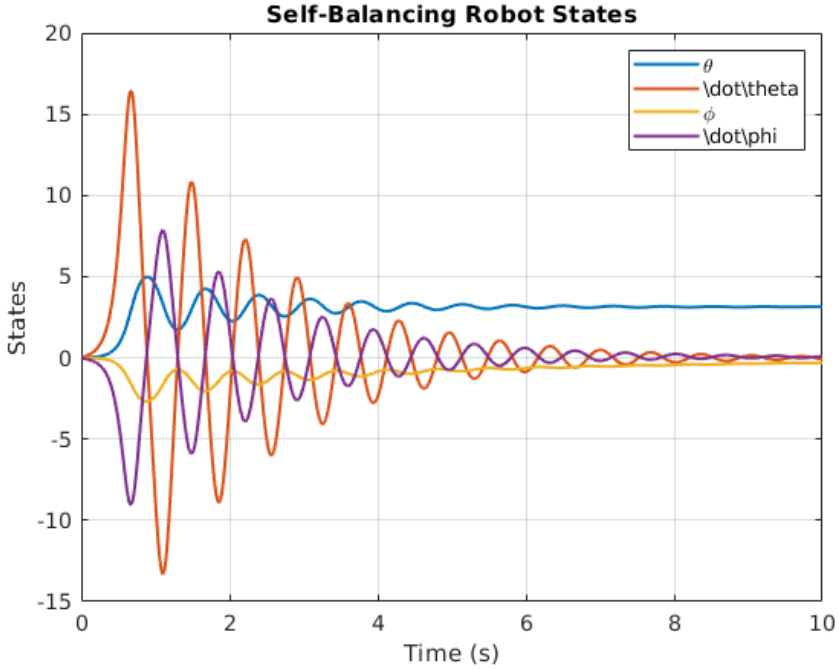

Figure 3: Time-response of robot motion.

and the output vector

$$y = \begin{bmatrix} \theta \\ \phi \end{bmatrix},$$

the transfer functions from wheel torque input u to the measurable outputs θ and ϕ were obtained using the `tf` command in MATLAB.

1. Transfer function $u \to \theta$

$$\frac{\Theta(s)}{U(s)} = \frac{4.258\,s + 0.7963}{s^3 + 1.313\,s^2 - 85.76\,s - 16.07}. \tag{17}$$

This transfer function is of third order because the wheel absolute angle ϕ corresponds to an uncontrollable and unobservable integrator mode for this particular input–output channel.

2. Transfer function $u \to \phi$

$$\frac{\Phi(s)}{U(s)} = \frac{-2.416\,s^2}{s^4 + 1.313\,s^3 - 85.76\,s^2 - 16.07\,s}. \tag{18}$$

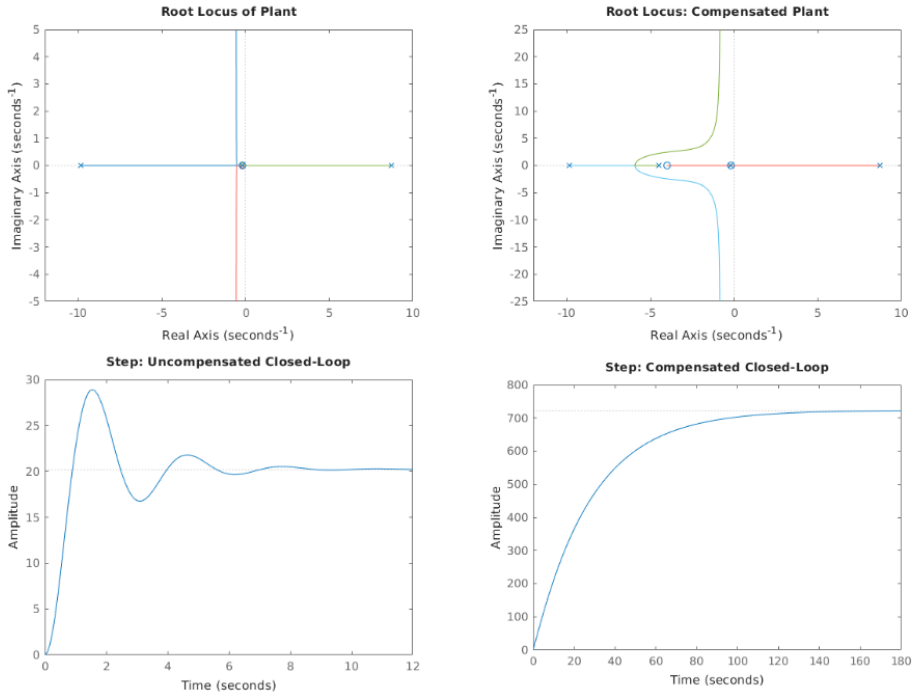

Figure 4: Enter Caption

2.2 Root–Locus Based Controller Design

A lead compensator was designed using the root–locus technique to stabilise the upright equilibrium. A desired dominant pole location was selected on the plant root locus, and the corresponding uncompensated gain was obtained using the magnitude condition. To reshape the locus and force it through the desired pole, a compensator zero was placed at $z_c = -4$, and the compensator pole p_c was computed using the angle–deficit rule. The compensated system gain was then obtained from the magnitude condition evaluated at the same pole location.

2.3 Full-State Feedback Control Using Pole Placement

To stabilise the self-balancing robot around the upright configuration, a full-state feedback controller of the form

$$u = -Kx$$

was designed. Since the full 4-state model is not fully controllable, a reduced 3-state subsystem

$$x_r = \begin{bmatrix} \theta \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix}$$

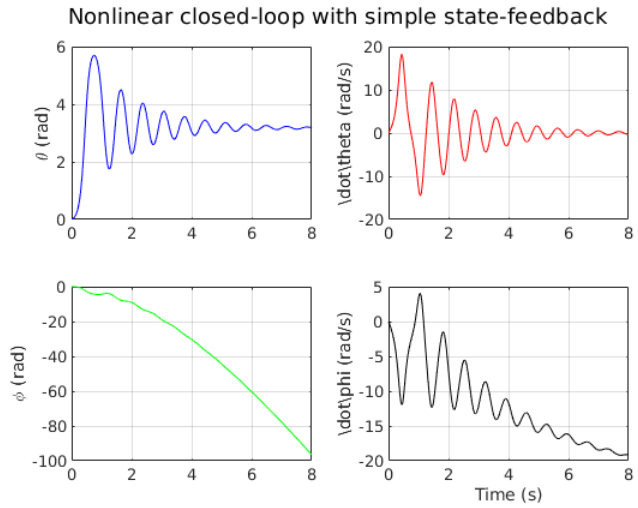

Figure 5: Time evolution of state variables

was extracted and used for control design. The matrices of the reduced linearised model are

$$A_r = \begin{bmatrix} 0 & 1 & 0 \\ a_{21} & a_{22} & a_{24} \\ a_{41} & a_{42} & a_{44} \end{bmatrix}, \qquad B_r = \begin{bmatrix} 0 \\ b_2 \\ b_4 \end{bmatrix},$$

which were shown to be controllable.

Desired closed-loop dynamics were imposed by pole placement with a set of moderately damped poles,

$$p_{\text{des}} = \{-2,\ -3,\ -4\},$$

giving the reduced-state gain

$$K_r = \texttt{place}(A_r, B_r, p_{\text{des}}).$$

This gain was mapped back into the full 4-state space as

$$K = [\, K_r(1) \;\; K_r(2) \;\; 0 \;\; K_r(3) \,].$$

The control input was limited using a realistic actuator saturation

$$u = \text{sat}(-Kx,\ u_{\max}), \qquad u_{\max} = 5 \text{ N·m},$$

and the nonlinear closed-loop system was simulated using `ode45`.

2.4 Comparison of PID and Full-State Feedback Controllers

Table 3 shows the performance of the PID controller (designed using root-locus) and the full-state feedback controller (designed using pole placement).

Table 3: Performance metrics of the two control schemes.

Controller	t_r (s)	M_p (%)	t_s (s)	SS value	SS error
Uncompensated PID (RL)	0.606	43.02	6.573	20.216	large
Compensated PID (RL)	61.246	0	109.057	723.019	very large
State Feedback	0.240	79.11	6.565	3.179 rad	3.179 rad

Nature of Control Action. The PID/lead compensator uses only the output angle θ. Its action is based on the error signal and it shapes a single feedback loop. It cannot directly control internal states, so its response is limited by plant dynamics.

The full-state feedback controller uses all states $[\theta,\ \dot{\theta},\ \phi,\ \dot{\phi}]^T$. It acts on each state individually, giving faster and more direct control over the system dynamics.

Numerical Comparison. The compensated PID removes overshoot but becomes very slow, with a rise time above 60 s. The state-feedback controller is much faster (0.24 s) but shows higher overshoot due to the aggressive pole placement and the lack of an integrator. The PID approach gives smoother but slower motion, while the state-feedback method gives high bandwidth and quick correction. The performance difference follows directly from the information available to each controller and the way they influence the system.

2.5 Full-state estimation (practical / reduced implementation)

The linearised plant with state $x = [\theta,\ \dot{\theta},\ \phi,\ \dot{\phi}]^T$ is not fully observable from a single output $y = \theta$ ($\text{rank}\big(\mathcal{O}(A, C)\big) = 3$). Consequently a classical 4-state Luenberger observer is not feasible using θ alone.

We therefore design an observer for the observable subsystem $[\,\theta,\ \dot{\theta},\ \dot{\phi}\,]$. The reduced matrices (A_r, B_r, C_r) are formed from the linearisation and the reduced observer gain L_r is computed by pole placement with fast stable poles (e.g. $-8, -12, -16$). The controller uses state-feedback $u = -K_{\text{full}}\hat{x}$ where $\hat{x} = [\hat{\theta},\ \hat{\dot{\theta}},\ 0,\ \widehat{\dot{\phi}}]^T$ (the absolute wheel angle ϕ is set to zero because $K_{\text{full}}(3) = 0$). The observer equations are simulated in cascade with the nonlinear plant and the estimator converges to the true observable states despite the unobservable integrator mode.

Design choices Observer poles are chosen 2–5× faster than controller poles to ensure rapid estimation while avoiding noise amplification. The reduced observer is sufficient because the controller does not rely on ϕ.

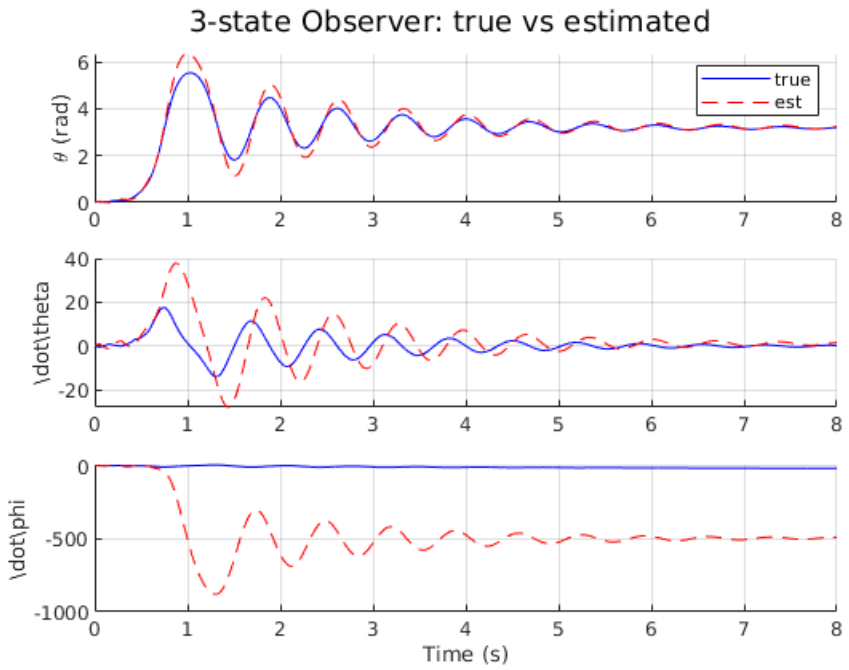

Figure 6: True vs estimated states

2.6 Reduced-Order Observer

A reduced-order observer is designed so that only the measured state θ is used, while the remaining three states are estimated. The system is partitioned into measured and unmeasured components, and the observer is built on the controllable subspace of $(A_{bb}^\top, A_{ab}^\top)$. Fast observer poles are assigned so that the estimated states converge quickly. The reconstructed full state is then used for state-feedback control. This reduced-order observer achieves the same stabilising behaviour while requiring only a single measurement.

3 Noise Injection, FFT Analysis, Filtering and Observer Performance

3.1 Noise Injection Using Simulink

Measurement noise was added using the built-in *Band-Limited White Noise* block in Simulink. The noisy measurement is

$$\tilde{\theta}(t) = \theta(t) + n(t), \qquad n(t) \sim \mathcal{N}(0, 0.01^2).$$

This noisy signal was supplied to both the full-order and minimum-order observers.

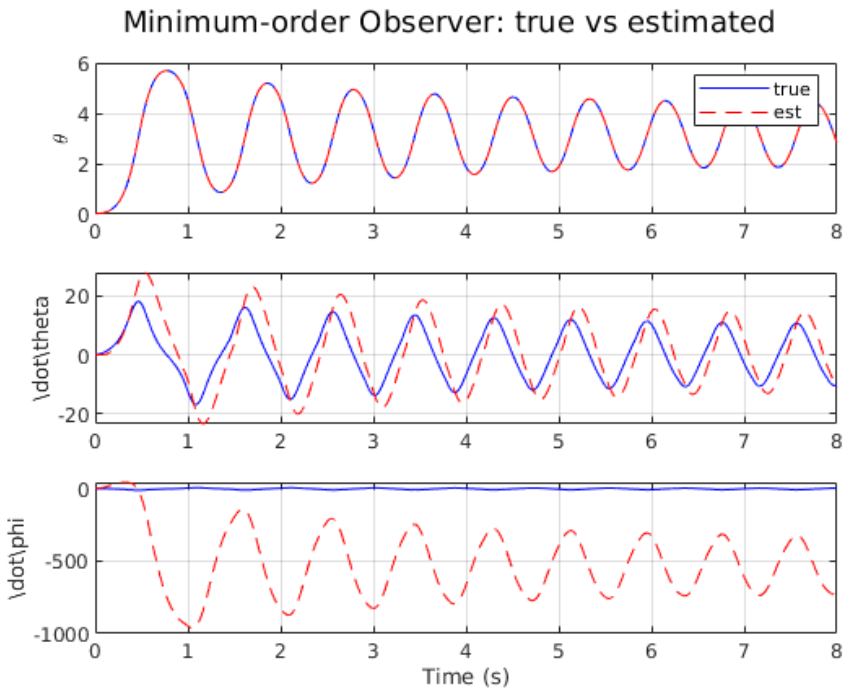

Figure 7: Reduced order observer

Observation: Both observers remained stable, but the full-order observer showed higher sensitivity to noise due to its innovation term.

3.2 FFT-Based Noise Analysis

The noisy output was analysed using MATLAB's FFT:

$$Y(f) = \text{FFT}(\tilde{\theta}(t)).$$

The single-sided magnitude spectrum revealed a dominant low-frequency component (pendulum dynamics) and broadband high-frequency noise.

Observation: High-frequency noise justified the use of a low-pass filter before observer correction.

3.3 Low-Pass Filtering

A second-order Butterworth low-pass filter with cutoff frequency $f_c = 5$ Hz was applied:

$$\theta_f(t) = \text{filtfilt}(b, a, \tilde{\theta}(t)).$$

Observation: Filtering effectively removed high-frequency noise with negligible phase delay.

3.4 Observer Performance with Filtered Output

Full-Order Observer Filtering significantly reduced estimation errors. The observer produced accurate and smooth state estimates.

Minimum-Order Observer The reduced-order observer showed better robustness to noise even without filtering, but filtering still improved smoothness and accuracy.

3.5 Impact on Feedback Control

- **Unfiltered noise:** Increased control effort and small torque oscillations.
- **Filtered output:** Smooth control input, faster settling, and improved state tracking.

3.6 Summary

- FFT confirmed that noise was concentrated at high frequencies.
- Butterworth filtering greatly enhanced observer performance.
- Full-order observer benefited the most from filtering.
- Overall closed-loop behaviour improved significantly with filtered measurements.

References

References

[1] University of Michigan, CTMS: Inverted Pendulum System Modeling (web tutorial). *Control Tutorials for MATLAB and Simulink (CTMS).* Available online (consulted for EOM forms and state-space conversion). `https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum`

[2] F. Jeremic, "Derivation of Equations of Motion for Inverted Pendulum", course notes (example derivation reference).

[3] C. T. Refvem (chapter), "Design, Modeling and Control of a Two-wheel Balancing Robot" (core.ac.uk PDF). (Used for two-DOF structure and parameter grouping.)

[4] H. Zhang et al., "Control Strategies for Two-Wheeled Self-Balancing...", *Robotics* (MDPI) — survey and modeling techniques.

[5] Markus Buchholz, “Dynamics Modelling and Simulation of Self-balancing Robot in C++”, Medium article (practical tutorial).

[6] “Enhanced Self Balancing Robot Simulation in Simulink”, YouTube tutorial (useful Simulink implementation patterns).