ME5659 Control Systems Engineering
Northeastern University | Spring 2026

Segway Controller Design
Term Project Report

Ahmad Hassan | NUID: 002560629
hassan.ahmad@northeastern.edu

Collaboration: I did not collaborate with anyone

Table of Contents

Problem Statement

System description

A Segway is a two-wheeled self-balancing personal transporter. The rider stands on a platform mounted above a single wheel axis, and because the platform is free to rotate about that axis, the system behaves as an inverted pendulum. It would tip over without active stabilization. Two motors drive the wheels; the controller's job is to command motor torque T(t) in a way that keeps the platform upright while still letting the rider travel forward and backward.

The rider's intent is conveyed through body posture. Leaning forward shifts the rider's mass by a horizontal offset d, which produces a gravitational toppling torque about the wheel axis. To prevent the fall, the controller accelerates the vehicle forward, driving the wheels beneath the tipping body and restoring balance. This accelerating motion is what makes the Segway travel. Returning to vertical (d = 0) leaves the vehicle coasting at constant speed, subject only to air drag; leaning backward creates the opposite imbalance and decelerates the vehicle. The whole design therefore depends on closed-loop feedback. Without a controller actively reading the tilt and commanding torque in real time, the vehicle would simply fall over.

Geometry and variables

Figure 1 shows the vehicle's critical dimensions and the two coordinates used throughout the report: the tilt angle θ measured from vertical, and the horizontal displacement x of the wheel axis.

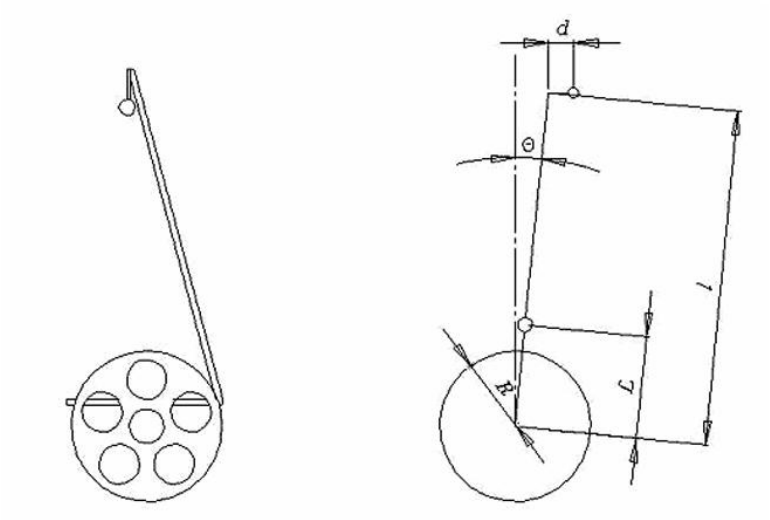


Figure 1: Segway vehicle geometry. θ is the tilt angle from vertical, d is the rider's horizontal mass offset from the wheel axis, R is the wheel radius, L is the distance from the wheel axis to the vehicle's center of mass, and ℓ is the distance from the wheel axis to the rider's center of mass.

Governing equations

Under the simplifying assumptions stated in the project brief, the dynamics reduce to two second-order ODEs — one for rotation about the wheel axis (Eq. I) and one for horizontal translation (Eq. II):

$$J\ddot{\theta}(t) = MgL\,\theta(t) + mg\ell\,\theta(t) - T(t) - c_1\dot{\theta}(t) + mg\,d(t) \qquad \text{(I)}$$

$$(M+m)\ddot{x}(t) = \frac{T(t)}{R} - c_2\dot{x}(t) \qquad \text{(II)}$$

The inputs are the motor torque T(t) (the control variable) and the rider imbalance d(t) (the disturbance). The outputs of interest are the tilt angle θ(t) and the vehicle speed $\dot{x}(t)$.

Project objectives

The project asks for the progression of increasingly capable controllers for this system:

- **Open-loop analysis** — confirm the plant is unstable and quantify why (Step 1)
- **State-space modeling** — reformulate the dynamics and verify controllability (Step 2)
- **PID design** — stabilize the system using only tilt measurements (Steps 3, 4)
- **Full-state feedback** — redesign with all four states measured, and study saturation and steady-state behavior (Steps 5, 6, 7)
- **Reduced-order design** — drop the integrator state and compare (Step 8)
- **Power analysis** — compare torque and power demands across controllers (Step 9)
- **Observer design** — estimate unmeasured states and close the loop using estimates (Step 10)

Each step is simulated in MATLAB and Simulink, and results are compared both against analytical predictions and across designs.

System figure and governing equations adapted from the ME 5659 Term Project brief (Northeastern University, Spring 2026)

Step 1: Laplace Representations and Open-Loop Stability

Main question: "Is this thing stable on its own?"
Before designing any controller, we need to know whether the Segway can stay upright without one. It can't — it's an inverted pendulum, so any small tilt is amplified by gravity. This step makes that rigorous by finding the transfer functions from torque and disturbance to the outputs, locating their poles on the s-plane, and confirming one pole lies in the right-half plane. That single RHP pole is what makes feedback control mandatory rather than optional.

1.1 Starting equations of motion

From the project statement:

$$J\ddot{\theta}(t) = MgL\,\theta(t) + mgl\,\theta(t) - T(t) - c_1\,\dot{\theta}(t) + mg\,d(t) \qquad -(I)$$

$$(M+m)\ddot{x}(t) = \frac{T(t)}{R} - c_2\dot{x}(t) \qquad -(II)$$

Inputs: torque $T(t)$(control) and rider imbalance $d(t)$(disturbance). Outputs of interest: tilt angle $\theta(t)$ and vehicle speed $\dot{x}(t)$.

1.2 Parameter values (NUID digit 9)

Symbol	Value	Units
R	0.35	m
L	0.55	m
ℓ	0.85	m
M	20	kg
m	90	kg
c_1	1	N·m·s/rad
c_2	10	N·s/m
g	9.81	m/s²
J = ML² + mℓ²	71.075	kg·m²

Per the professor's clarification, J is computed by treating both the vehicle mass and the rider mass as point masses at their respective distances from the wheel axis:

$$J = ML^2 + m\ell^2 = 20(0.55)^2 + 90(0.85)^2 = 6.050 + 65.025 = 71.075\ kgm^2$$

Useful pre-computed constants:

- $MgL = 107.910N \cdot m/rad$
- $mg\ell = 750.465\ N.m/rad$
- $MgL + mg\ell = 858.375\ N.m/rad$
- $mg = 882.900\ N/m$
- $M + m = 110\ kg$

1.3 Laplace transform of (I) — θ dynamics

Taking Laplace transforms of (I) with zero initial conditions $\theta(0) = \dot{\theta}(0) = 0$:

$$Js^2\Theta(s) = (MgL + mg\ell)\Theta(s) - T(s) - c_1 s\,\Theta(s) + mg\,D(s)$$

Collecting $\Theta(s)$ on the left:

$$Js^2 + c_1 s - (MgL + mg\ell)\,]\,\Theta(s) = -T(s) + mg\,D(s)$$

Define the θ-subsystem characteristic polynomial:

$$\boxed{\Delta_\theta(s) \;=\; Js^2 + c_1 s - (MgL + mg\ell) \;=\; 71.075\,s^2 + s - 858.375}$$

The two transfer functions are:

$$\boxed{\left.\frac{\Theta(s)}{T(s)}\right|_{D=0} = \frac{-1}{\Delta_\theta(s)} = \frac{-1}{71.075\,s^2 + s - 858.375}}$$

$$\boxed{\left.\frac{\Theta(s)}{D(s)}\right|_{T=0} = \frac{mg}{\Delta_\theta(s)} = \frac{882.90}{71.075\,s^2 + s - 858.375}}$$

*In MATLAB, these two transfer functions are built with the tf command, e.g. G_theta_T = -1 / (J*s^2 + c1*s - (M*g*L + m*g*ell)) after defining s = tf('s').*

Sign check: a positive torque T decelerates the tilt (hence the minus sign), while a positive d (rider leans forward) creates a toppling moment that increases θ.

1.4 Laplace transform of (II) — ẋ dynamics

Taking Laplace of (II) with zero initial conditions:

$$(M + m)\,s \cdot \dot{X}(s) = \frac{T(s)}{R} - c_2\,\dot{X}(s)$$

where $\dot{X}(s) := \mathcal{L}\{\dot{x}(t)\} = sX(s)$. Collecting:

$$[\,(M + m)\,s + c_2\,]\,\dot{X}(s) = \frac{T(s)}{R}$$

Define the ẋ-subsystem characteristic polynomial:

$$\boxed{\Delta_{\dot{x}}(s) \;=\; (M + m)\,s + c_2 \;=\; 110\,s + 10}$$

So:

$$\boxed{\left.\frac{\dot{X}(s)}{T(s)}\right|_{D=0} = \frac{1}{R[(M + m)\,s + c_2]} = \frac{1}{0.35\,(110\,s + 10)} = \frac{1}{38.5\,s + 3.5}}$$

$$\boxed{\left.\frac{\dot{X}(s)}{D(s)}\right|_{T=0} = 0}$$

In MATLAB, this is similarly constructed with tf: G_xdot_T = 1 / (R((M+m)*s + c2)).*

1.5 Physical interpretation of $\dot{X}(s)/D(s) = 0$

Equation (II) contains *only* T on its RHS, neither θ nor d appears. So, in the **open loop** (no feedback controller), rider lean has no direct effect on horizontal acceleration. Leaning only tilts the body; the vehicle only accelerates if a torque T is applied.
The Segway's ability to move in response to rider lean is therefore a **closed-loop** phenomenon: the controller senses θ, generates T, and T then drives $\dot{x}$ via Eq. (II).

1.6 Transfer-function matrix form

Packing the four transfer functions:

$$\begin{bmatrix} \Theta(s) \\ \dot{X}(s) \end{bmatrix} = \underbrace{\begin{bmatrix} \dfrac{-1}{\Delta_\theta(s)} & \dfrac{mg}{\Delta_\theta(s)} \\ \dfrac{1}{R\,\Delta_{\dot{x}}(s)} & 0 \end{bmatrix}}_{G(s)} \begin{bmatrix} T(s) \\ D(s) \end{bmatrix}$$

$G(s)$ is block-triangular — the lower-right zero is exactly the open-loop decoupling of 1.5.

1.7 Pole analysis

Poles of θ subsystem — roots of $\Delta_\theta(s) = 71.075\,s^2 + s - 858.375 = 0$:

$$s = \frac{-1 \pm \sqrt{1 + 4 \cdot 71.075 \cdot 858.375}}{2 \cdot 71.075} = \frac{-1 \pm \sqrt{244{,}037.01}}{142.15} = \frac{-1 \pm 494.00}{142.15}$$

$$\boxed{s_1 = +3.468 \quad \text{(RHP — UNSTABLE)}, \qquad s_2 = -3.482}$$

Pole of ẋ subsystem — root of $\Delta_{\dot{x}}(s) = 110\,s + 10 = 0$:

$$\boxed{s_3 = -\frac{c_2}{M+m} = -\frac{10}{110} = -0.0909 \quad \text{(LHP — stable)}}$$

The full set of open-loop poles for the combined system is $\{+3.468,\ -3.482, -0.0909\}$.

*In MATLAB, the poles are computed using the pole command on each transfer function: pole(G_theta_T) and pole(G_xdot_T). Equivalently, roots([J c1 -(M*g*L + m*g*ell)]) applied directly to the characteristic-polynomial coefficients returns the same values.*

1.8 Stability verdict

A linear system is BIBO/asymptotically stable iff **every** pole of its transfer function lies in the open left-half plane. Here:

- $s_1 = +3.468$ is in the **right-half plane** → the θ dynamics are **unstable**.
- s_2, s_3 are both LHP but cannot compensate for the unstable mode.

Therefore, the open-loop Segway plant is unstable.

Physical reason: equation (I) says $\ddot{\theta} \propto +(MgL + mg\ell)\,\theta$, i.e. gravity is *destabilizing*. Any small positive θ produces a positive angular acceleration that further increases θ. This is the classical inverted-pendulum

instability. The damping $c_1 > 0$ is not enough to overcome the gravity torque, which is why one root crosses into the RHP.

Note also that $s_1 \approx -s_2$ with a small offset caused by the damping c_1. Without damping ($c_1 = 0$), the roots would be exactly $\pm\sqrt{(MgL + mg\ell)/J} = \pm 3.475$, symmetric about the imaginary axis, the textbook undamped-inverted-pendulum pair.

Closed-loop stabilization via T is therefore **mandatory**.

1.9 Figure

Pole–zero map of the open-loop plant, showing all three poles on the s-plane. The pole at $s = +3.468$ lies in the right-half plane, confirming instability visually.

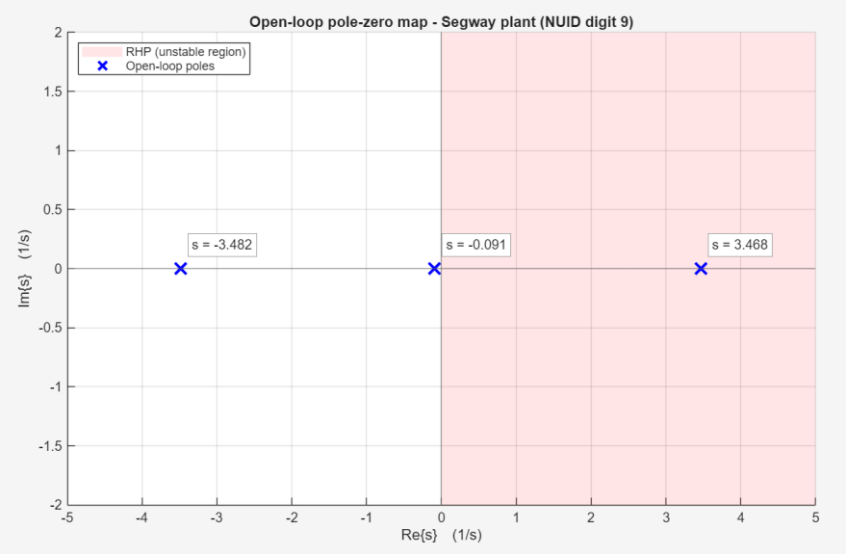


Figure 2: Open-loop pole-zero map

1.10 Summary table

Transfer function	Expression	Poles	Stable?
Θ/T	$-1/\Delta_\theta$	+3.468, −3.482	✗ (RHP pole)
Θ/D	mg/Δ_θ	+3.468, −3.482	✗ (RHP pole)
$\dot{X}/T$	$1/[R\Delta_{\dot{x}}]$	−0.0909	✓
$\dot{X}/D$	0	—	(trivial)

Overall plant: unstable — stabilization requires feedback control

Step 2: State-Space Form, Controllability, Output Controllability

Main question: Can a single motor actually control everything we care about?
With stability ruled out, we set up the bookkeeping for control design by rewriting the equations of motion in state-space form. The four states are ∫θ, θ, θ̇, and ẋ. The integral of tilt is included up front so that Step 3's PID can be implemented as standard state feedback. We then run two tests: state controllability (can the torque steer the full 4-dimensional state?) and output controllability (can it independently steer both θ and ẋ?). Both come back positive, so control design is on solid footing.

2.1 State definition

Per the project statement, the state vector is:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \int_0^t \theta(\tau)\, d\tau \\ \theta \\ \dot{\theta} \\ \dot{x} \end{bmatrix}$$

The input is $u = T$ (motor torque) and the disturbance is d (rider imbalance).

Why the integrator state $x_1 = \int \theta\, dt$? Step 3 will design a PID controller of the form $T = -[\,k_1, k_2, k_3, 0]x = -k_1 \int \theta\, dt - k_2\, \theta - k_3\, \dot{\theta}$. The integral-of-θ term (the $k_1$term) requires $\int \theta\, dt$ to be a state. Including it here lets us use standard state-feedback tools ($T = -Kx$) to realize a PID controller, instead of having to treat the integrator separately.

2.2 Deriving each state derivative

From the defining relations:

$$\dot{x}_1 = \frac{d}{dt} \int_0^t \theta(\tau)\, d\tau = \theta = x_2$$
$$\dot{x}_2 = \dot{\theta} = x_3$$

From EoM (I), $J\ddot{\theta} = (MgL + mg\ell)\theta - T - c_1\dot{\theta} + mg\, d$, so:

$$\dot{x}_3 = \ddot{\theta} = \frac{MgL + mg\ell}{J}\, x_2 \; - \; \frac{c_1}{J}\, x_3 \; - \; \frac{1}{J}\, T \; + \; \frac{mg}{J}\, d$$

From EoM (II), $(M + m)\ddot{x} = T/R - c_2\dot{x}$, so:

$$\dot{x}_4 = \ddot{x} = -\frac{c_2}{M + m}\, x_4 \; + \; \frac{1}{R(M + m)}\, T$$

2.3 State-space form

Assembling $\dot{x} = Ax + Bu + Fd$:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \dfrac{MgL + mg\ell}{J} & -\dfrac{c_1}{J} & 0 \\ 0 & 0 & 0 & -\dfrac{c_2}{M + m} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{1}{J} \\ \dfrac{1}{R(M + m)} \end{bmatrix}, \qquad F = \begin{bmatrix} 0 \\ 0 \\ \dfrac{mg}{J} \\ 0 \end{bmatrix}$$

Substituting numerical values ($J = 71.075$, $M + m = 110$, $R(M + m) = 38.5$, $MgL + mg\ell = 858.375$, $mg = 882.9$):

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 12.0770 & -0.01407 & 0 \\ 0 & 0 & 0 & -0.09091 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ -0.01407 \\ 0.02597 \end{bmatrix}, \quad F = \begin{bmatrix} 0 \\ 0 \\ 12.4221 \\ 0 \end{bmatrix}$$

The outputs of interest from Step 1 are θ and $\dot{x}$, so:

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Structural observation: the first column of A is all zeros. This is the signature of the integrator state x_1. No other state's derivative depends on x_1, which matches physics (the integral of tilt doesn't feed back into anything). Similarly, the x_4 row is decoupled from (x_2, x_3), the same open-loop decoupling we identified in 1.5 of Step 1.

2.4 State controllability

A pair (A, B) is **state controllable** iff the Kalman controllability matrix

$$\mathcal{C} = [B \quad AB \quad A^2B \quad A^3B]$$

has full rank $n = 4$.

Computed numerically:

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & -0.01407 & 0.000198 \\ 0 & -0.01407 & 0.000198 & -0.1699 \\ -0.01407 & 0.000198 & -0.1699 & 0.00479 \\ 0.02597 & -0.00236 & 0.000215 & -0.0000195 \end{bmatrix}$$

$$\boxed{\text{rank}(\mathcal{C}) = 4 \implies (A, B) \text{ is fully state controllable.}}$$

In MATLAB, the controllability matrix is built with ctrb(A,B) and its rank checked with rank(ctrb(A,B)).

Interpretation. A single scalar input T can steer the entire 4-dimensional state to any target in finite time. This is non-trivial: a staircase look at $\mathcal{C}$ shows that each successive power of A "propagates" the effect of B up through the state hierarchy — first into x_3(column 1), then x_2(column 2), then x_1(column 3). Since x_4 was already touched directly by B, all four states are reachable.

2.5 Output controllability

*Definition (Ogata, *Modern Control Engineering* 5th ed.; also Wikipedia, "Controllability").*
The system (A, B, C, D) is **output controllable** iff the output controllability matrix

$$\mathcal{C}_o = [CB \quad CAB \quad CA^2B \quad CA^3B \quad D]$$

has full **row** rank equal to the number of outputs m. Since $D = 0$ here, the D column is dropped (it can't change the rank).

This is distinct from state controllability: state controllability asks whether we can steer the full n-dim state, while output controllability asks the (often weaker) question of whether we can steer the m-dim output. The two are independent. A system can satisfy one without the other.

Computation. With $m = 2$ outputs (θ and $\dot{x}$):

$$\mathcal{C}_o = \begin{bmatrix} 0 & -0.01407 & 0.000198 & -0.1699 \\ 0.02597 & -0.00236 & 0.000215 & -0.0000195 \end{bmatrix}$$

This is a 2×4 matrix. We need rank 2:

$$\boxed{\text{rank}(\mathcal{C}_o) = 2 \implies \text{the system is output controllable.}}$$

Inspection confirms this: the first column $[0, 0.02597]^T$ and any of the other columns (e.g. $[-0.01407, -0.00236]^T$) are linearly independent, giving rank 2 immediately.

*In MATLAB, this is computed as C_o = C * ctrb(A,B) (since by definition $\mathcal{C}_o = C\,[\,B \;\; AB\, A^2B\, A^3B\,] = C \cdot \text{ctrb}(A,B)$), followed by rank(C_o).*

Interpretation. Both tilt angle and vehicle speed can be independently driven to any desired values by appropriate choice of the torque input $T(t)$. This is consistent with physical intuition: the motor torque both influences the tilt (through the reaction torque on the body) and pushes the vehicle horizontally (through the wheel contact), so both outputs are reachable.

2.6 Summary

Property	Test	Result	Conclusion
State controllability	$rank\,[\,B\;AB\;A^2B\;A^3B\,] = n$	$4 = 4$	✓
Output controllability	$rank\,[\,CB\;CAB\;CA^2B\;CA^3B\,] = m$	$2 = 2$	✓

Both properties hold. The system is fully controllable from T, and both outputs $\theta, \dot{x}$ are independently steerable.

Step 3: PID Controller Design and Simulink Simulation

Main question: Can we stabilize the Segway using only tilt measurements?
The first controller has to work with a real-world constraint: there's no sensor for ẋ. The only measurement is θ, from which θ̇ and ∫θ can be computed. That's exactly the information a classical PID uses. We place three closed-loop poles (the fourth is locked by the ẋ dynamics), solve for the gains, and verify in Simulink that a small initial tilt decays smoothly to zero.

3.1 Closed-loop structure

The controller is $T = -Kx$ with the PID-structured gain

$$K = [k_1 \quad k_2 \quad k_3 \quad 0]$$

so $T = -k_1 \int \theta \, dt - k_2\theta - k_3\dot{\theta}$. The zero in the fourth slot enforces the constraint that $\dot{x}$ is not available for feedback.

The closed-loop state matrix is $A - BK$. Computing:

$$A - BK = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{k_1}{J} & \frac{(MgL + mg\ell) + k_2}{J} & \frac{k_3 - c_1}{J} & 0 \\ -\frac{k_1}{R(M+m)} & -\frac{k_2}{R(M+m)} & -\frac{k_3}{R(M+m)} & -\frac{c_2}{M+m} \end{bmatrix}$$

Key structural observation. The fourth column of $A - BK$ is $\begin{bmatrix} 0 & 0 & 0 & -\frac{c_2}{M+m} \end{bmatrix}^T$, so the matrix is block upper-triangular. The characteristic polynomial therefore factors:

$$\det(sI - (A - BK)) = \underbrace{\left(s + \frac{c_2}{M+m}\right)}_{\text{fixed } \ddot{x} \text{ pole}} \cdot \underbrace{\det(sI - (A_{3\times3} - B_{3\times3}K_{3\times3}))}_{\text{3 movable poles}}$$

This means we can place only **3 poles** with our 3 free gains (k_1, k_2, k_3); the 4th pole is locked at $-c_2/(M+m) = -0.0909$, the damping of the horizontal subsystem, unaffected by feedback on θ.

3.2 Pole placement on the 3-state subsystem

The upper-left 3×3 block in companion form is

$$A_{3\times3} - B_{3\times3}K_{3\times3} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{k_1}{J} & \frac{(MgL + mg\ell) + k_2}{J} & \frac{k_3 - c_1}{J} \end{bmatrix}$$

with characteristic polynomial (computed via cofactor expansion along the first column):

$$p_{cl}(s) = s^3 - \frac{k_3 - c_1}{J}s^2 - \frac{(MgL + mg\ell) + k_2}{J}s - \frac{k_1}{J}$$

Desired poles. I choose a dominant complex pair and a faster real pole:

$$poles = \{-3 + 3j, -3 - 3j, -6\}$$

giving a dominant-pair real part of -3 (natural frequency $\omega_n = 3\sqrt{2} \approx 4.24$rad/s, damping ratio $\zeta = 1/\sqrt{2} \approx 0.707$) and a settling time of roughly $4/3 \approx 1.3$s. This is a reasonable default: fast enough to demonstrate stability clearly, not so aggressive that we need enormous gains.

The desired characteristic polynomial is

$$p_{des}(s) = (s + 3 - 3j)(s + 3 + 3j)(s + 6) = (s^2 + 6s + 18)(s + 6) = s^3 + 12s^2 + 54s + 108$$

Matching coefficients with $p_{cl}(s)$ and solving:

$$\frac{c_1 - k_3}{J} = 12 \quad \Rightarrow \; k_3 = c_1 - 12J = 1 - 12(71.075) = -851.9$$

$$-\frac{(MgL + mg\ell) + k_2}{J} = 54 \quad \Rightarrow \; k_2 = -54J - (MgL + mg\ell) = -4696.4$$

$$-\frac{k_1}{J} = 108 \quad \Rightarrow \; k_1 = -108J = -7676.1$$

$$\boxed{K = [-7676.1 \quad -4696.4 \quad -851.9 \quad 0]}$$

In MATLAB, these gains can be computed analytically as above or equivalently using `place(A(1:3,1:3), B(1:3), [-3+3i, -3-3i, -6])` on the 3-state subsystem, which returns the same $K_{3\times3}$.

Verifying the full 4×4 closed-loop spectrum (via eig(A - B*K)):

$$\text{eig}(A - BK) = \{-3 \pm 3j, \; -6, \; -0.0909\}$$

All four eigenvalues are in the LHP → the closed-loop is asymptotically stable. ✓

3.3 Interpretation of gain signs

All three gains are **negative**. This is not an error as it follows from the sign of the control coefficient in the θ equation.

From EoM (I), $J\ddot{\theta} = (MgL + mg\ell)\theta - T - c_1\dot{\theta} + mgd$, the torque T appears with a *minus* sign. So, to apply a **restoring** torque when $\theta > 0$ (which requires $\ddot{\theta} < 0$), we need $T > 0$. Using $T = -Kx$, when $\theta = x_2 > 0$ we need $-k_2 x_2 > 0$, hence $k_2 < 0$. Same logic for the other components.

In classical PID notation with the standard sign convention ($T = K_p e + K_i \int e\, dt + K_d \dot{e}$ where $e = \theta_{\text{set}} - \theta = -\theta$), our gains correspond to $K_p = |\, k_2 \,| = 4696.4$, $K_i = |\, k_1 \,| = 7676.1$, $K_d = |\, k_3 \,| = 851.9$, all positive as expected.

The magnitudes are large because the destabilizing gravity term $(MgL + mg\ell) = 858.4$N·m/rad is itself large and the proportional gain must overcome it.

3.4 Simulink model

The Simulink diagram uses one State-Space block for the plant with input [T; d] and a Gain block for the controller. Build it as follows:

1. **Plant — State-Space block**
   - A: the 4×4 A from step 2.3
   - B: [B, F] (i.e. a 4×2 matrix stacking the control and disturbance columns side-by-side)
   - C: I_4(identity — output the full state so the controller can feed it back)
   - D: $0_{4\times2}$
   - Initial conditions: [0; 0.05; 0; 0](a $\theta(0) \approx 2.86°$ tilt, zero other states)
2. **Controller — Gain block** with value -K = [7676.1, 4696.4, 851.9, 0], set to "Matrix(K*u)" mode; takes the state vector as input, outputs scalar T.
3. **Disturbance** — Constant block set to 0 (Step 3 demonstrates stability; Step 4 will replace this with the trapezoidal $d(t)$).
4. **Mux** — 2-input Mux combines T and d into the 2-vector expected by the plant's input port.

5. **Scopes** — tap the state vector to extract θ(2nd component), $\dot{x}$(4th component), and the controller output T.

Block-diagram layout:

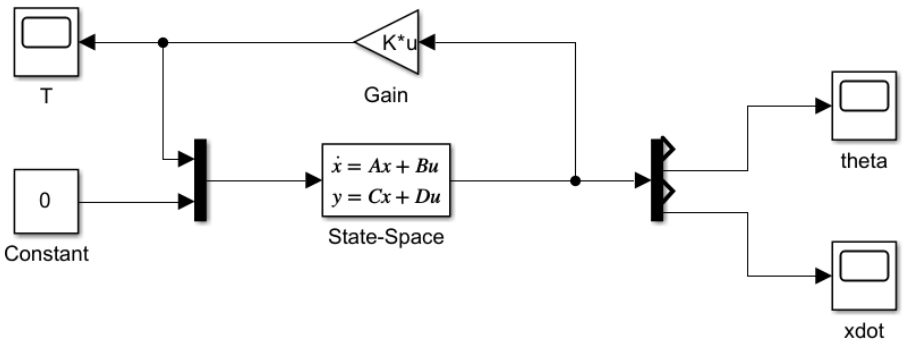


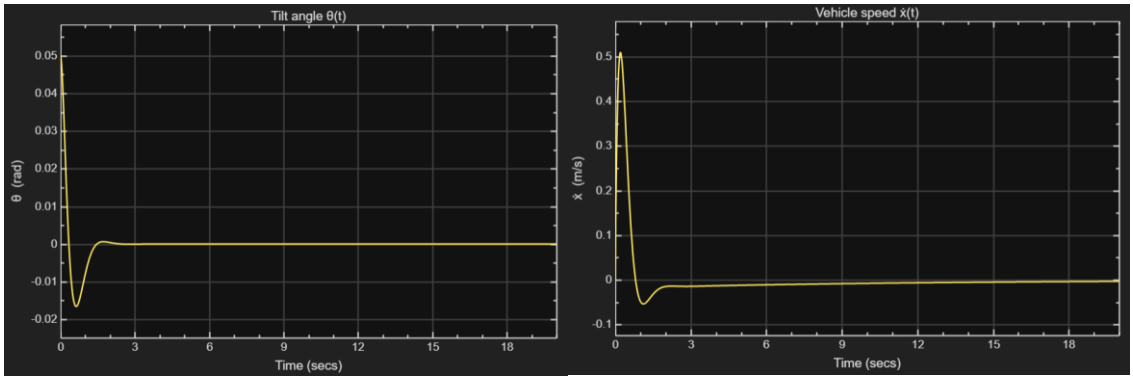
Figure 3: Simulink Model for Step 3

3.5 Simulation results

With $\theta(0) = 0.05$ rad and zero disturbance, the closed-loop simulation produces:

- $\theta(t)$ starts at 0.05rad ($\approx 2.86°$), undershoots briefly to ≈ -0.017rad around $t \approx 0.7$s, and settles to zero by $t \approx 2$s. The small overshoot is the signature of the dominant pair $-3 \pm 3j$(damping ratio $\zeta = 1/\sqrt{2} \approx 0.707$, theoretical peak overshoot ~4%).
- $\dot{x}(t)$ peaks at ≈ 0.5m/s near $t \approx 0.2$s — this is the side-effect of the restoring torque, which accelerates the vehicle horizontally — then undershoots slightly and decays slowly toward zero. The long tail reflects the uncontrolled pole at -0.0909 (time constant ≈ 11s).
- $T(t)$ peaks at ≈ 250N·m at $t = 0$, undershoots to ≈ -50N·m, and settles within ≈ 2s. The initial peak is dominated by the proportional term: $|k_2| \cdot \theta(0) = 4696.4 \times 0.05 \approx 235$N·m, which accounts for most of the observed 250 N·m.

Both outputs decay to zero → closed-loop system is stable, confirming the pole-placement design.



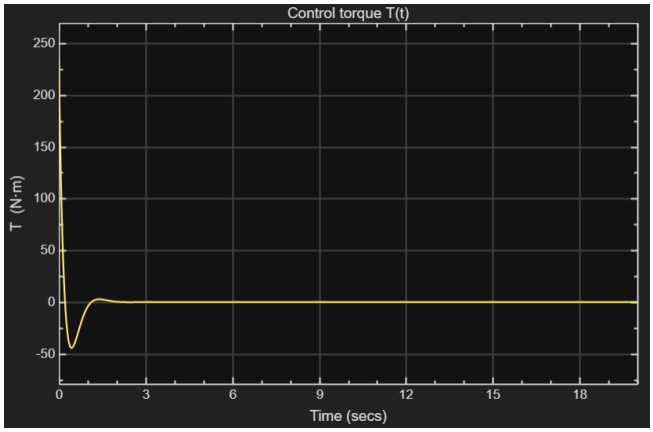


Figure 4: Scope output for tilt angle, vehicle speed and control torque (Step 3)

Remark on time scales. θ settles in ~2 s, but ẋ takes much longer. Physically, the controller's job is to keep θ at zero, and it succeeds quickly. The side-effect — horizontal acceleration generated by the restoring torque — then has to dissipate through the slow air-drag term $c_2/(M+m)$, which is why $\dot{x}$ has a long exponential tail.

Step 4: Trapezoidal Disturbance and Velocity Profile

Main question: Does the PID behave sensibly when the rider actually leans?
A real Segway moves because the rider leans, so we drive the closed-loop system with a trapezoidal d(t) that mimics a realistic lean cycle which is stand still, lean forward, hold, return to vertical. The lean amplitude is chosen using a static-gain calculation that keeps ẋ safely under the 3 m/s bound. We then check whether both constraints (|θ| < 10°, ẋ < 3 m/s) are met throughout the ride.

4.1 Designing the disturbance profile

The rider's behavior is modeled by a trapezoidal $d(t)$:

- $t \in [0,1)$: $d = 0$ (rider standing still)
- $t \in [1,2)$: d ramps linearly from 0 to d_{peak} (rider begins to lean forward)
- $t \in [2,8)$: $d = d_{peak}$ (rider holds the lean)
- $t \in [8,9)$: d ramps linearly from d_{peak} back to 0 (rider returns to upright)
- $t \geq 9$: $d = 0$

4.2 Choosing the peak amplitude

Per the problem statement, d is in meters and is expected to be in the cm range. The constraint $|\dot{x}| < 3$ m/s gives us a useful upper bound on d_{peak}.

Static analysis. With the PID controller from step 3 closing the loop, consider the system at steady state under a constant disturbance d_{ss}. Setting $\ddot{\theta} = \dot{\theta} = \theta = 0$ in Eq. (I) (the integral action drives $\theta \to 0$):

$$T_{ss} = mg\, d_{ss}$$

Setting $\ddot{x} = 0$ in Eq. (II):

$$\dot{x}_{ss} = \frac{T_{ss}}{R\, c_2} = \frac{mg\, d_{ss}}{R\, c_2}$$

With the numerical values, the static gain from d to $\dot{x}$ is:

$$\frac{\dot{x}_{ss}}{d_{ss}} = \frac{882.9}{0.35 \cdot 10} = 252.26\ s^{-1}$$

To respect the speed constraint $|\dot{x}| < 3$m/s, we therefore need

$$|d_{peak}| \lesssim \frac{3}{252.26} \approx 0.012\ \text{m}\ (\approx 1.2\ \text{cm})$$

Choice: $d_{peak} = 0.01$m (1 cm), safely below the static bound and in the middle of the expected cm range.

4.3 Building the disturbance in Simulink

The Constant 0 block from Step 3 was replaced with a Signal Editor block configured with six points and linear interpolation:

Time (s)	d (m)
0	0
1	0
2	0.01
8	0.01
9	0
15	0

The Signal Editor output was wired to the Mux input for the disturbance channel. Everything else (plant, controller, Mux, feedback wiring) remained identical to the Step 3 model. The initial condition on the State-Space block was changed to [0; 0; 0; 0] so the vehicle starts from rest.

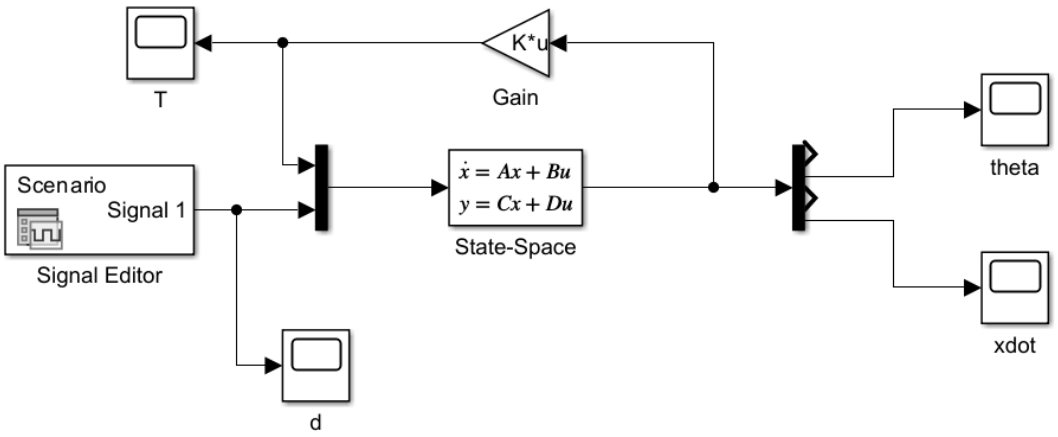


Figure 5: Simulink Model for Step 4

4.4 Simulation results

Simulation duration: 0 to 20s. Key traces observed:

- $d(t)$— the trapezoidal profile defined above, peak at 0.01m during $t \in [2,8]$.
- $\theta(t)$— stayed very close to zero throughout, with small transients of magnitude $\approx 0.066°$ only at the four "corners" of the trapezoid (the ramp starts and ends, where $\dot{d}$ changes). This is the integral action of the PID working as designed: the $k_1 \int \theta \, dt$ term prevents any persistent tilt.
- $\dot{x}(t)$— built up slowly during the $d > 0$ phase (the vehicle accelerated while the rider leaned), reached a peak of ≈ 1.17m/s near the end of the hold ($t \approx 8$ s), then decayed slowly back toward zero after $d$returned to zero.
- $T(t)$— followed a near-trapezoidal shape matching $d(t)$, with steady-state value $T_{ss} = mg\, d_{peak} = 8.83$N·m during the hold and small transient spikes at the corners (peak≈ 10.4 N·m).

Constraint check:

Quantity	Observed peak	Constraint	Margin
$\| \boldsymbol{\theta} \|$	$0.066°$	$\leq 10°$	✓(huge)
$\| \dot{\boldsymbol{x}} \|$	1.17m/s	$\lesssim$ 3m/s	✓

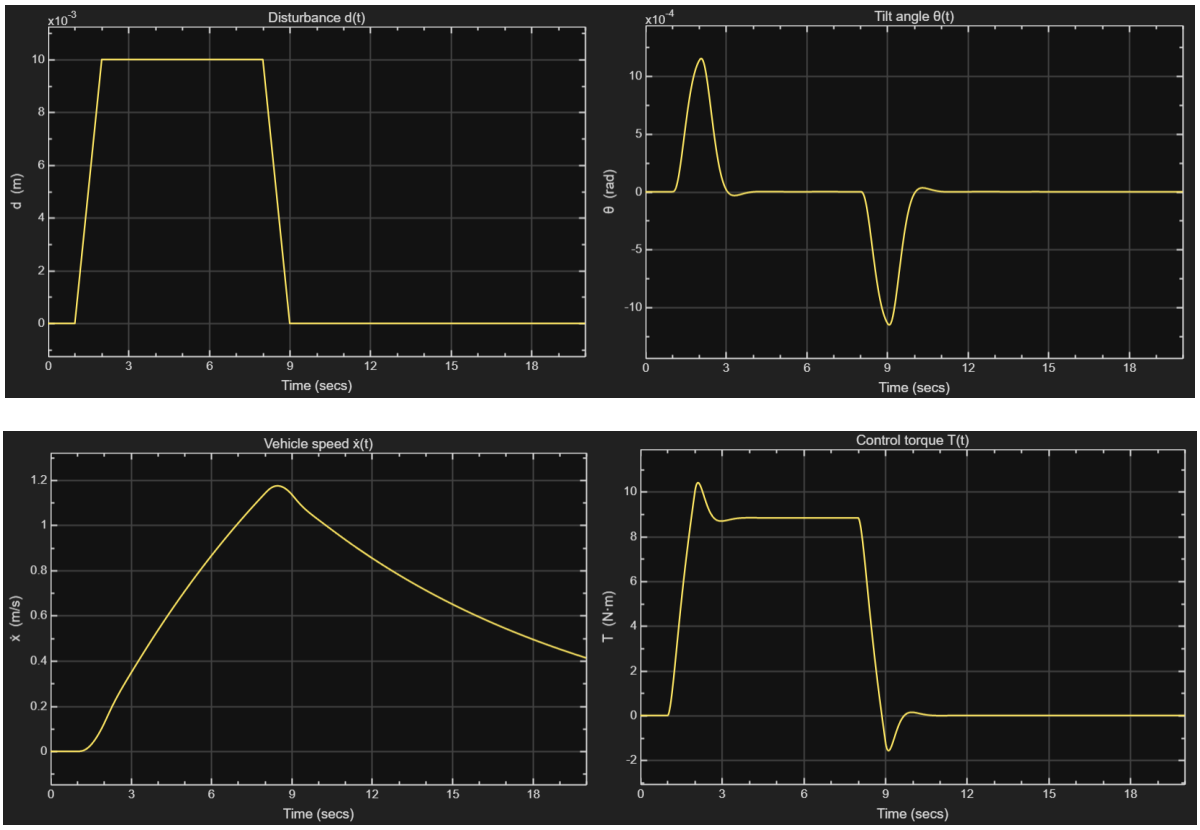


Figure 6: Scope output for disturbance, tilt angle, vehicle speed and control torque (Step 4)

4.5 Discussion

The first thing to notice is that ẋ never reaches the predicted steady-state value. Step 4.2 said a 1 cm lean should give ẋ_ss = 2.52 m/s, but the simulation only got to about 1.17 m/s during the hold. The

reason is the slow uncontrolled pole at −0.0909, which has a time constant of about 11 seconds. The hold phase only lasts 6 seconds, so ẋ has been rising for roughly half a time constant by the time the rider starts straightening up, not nearly long enough to settle. As a rough check, a first-order response would reach 1 − e^(−6/11) ≈ 42% of its asymptote in 6 seconds, which gives 0.42 × 2.52 ≈ 1.06 m/s. The simulation peak is a bit higher than this because the ramps mean torque is being applied for slightly longer than just the hold.

The other thing worth noting is how little θ moves. The disturbance torque reaches mg·d_peak = 8.83 N·m at the peak of the lean, but θ never exceeds 0.066°. This is exactly what the integral action in the PID is supposed to do. The ∫θ term in the controller drives the steady-state tilt error to zero under any constant disturbance, and the dominant poles at −3 ± 3j keep the transients well-damped.

It's also worth tying the simulation back to what a real Segway does. The trapezoidal d(t) mirrors a real rider's sequence which is lean forward to accelerate, hold the lean while moving, return to upright to stop. But the vehicle doesn't actually stop the moment the rider straightens up: ẋ is still about 0.4 m/s at t = 20 s, more than 10 seconds after the disturbance ended. That residual speed is the same slow drag pole showing up again. In a real Segway, the controller would presumably use a brake or a reverse-lean phase to bleed off the speed faster, but with this simple PID and aerodynamic drag as the only decelerating force, the vehicle just coasts down.

Step 5: Full-State Feedback via Pole Placement

Main question: What changes if we also have a wheel-speed sensor?
Now all four states are measurable, so the controller can react directly to ẋ instead of inferring its effect through θ. We redesign via pole placement with a 1-second settling-time spec and rerun the Step 4 disturbance. The comparison shows access to ẋ makes the vehicle track speed much better, but it also causes the controller to deliberately tilt the body during transients, producing larger θ swings than the PID.

5.1 Setup

With $\dot{x}$ now measurable, the controller is $T = -Kx$ with all four gains free. Step 2 showed (A, B) is fully controllable, so the closed-loop eigenvalues of $A - BK$ can be placed anywhere.

5.2 Desired pole locations

The problem asks for dominant eigenvalues with $t_s \approx 1$s. Using $t_s = 4/\sigma$, the dominant real part is $\sigma = 4$.

- **Dominant pair:** $-4 \pm 4j$ (natural frequency $\omega_n = 4\sqrt{2} \approx 5.66$ rad/s, damping $\zeta = 1/\sqrt{2} \approx 0.707$, ~4% overshoot)
- **Non-dominant:** -5 and -6

The non-dominant poles are kept close to the dominant pair rather than placed far left, because the gain magnitudes grow rapidly as poles move deeper into the LHP.

5.3 Computing K

Pole placement yields:

$$K \approx [-750{,}552 \quad -15{,}914 \quad 57{,}433 \quad 31{,}838]$$

In MATLAB, the gains are computed with K = place(A, B, [-4+4i, -4-4i, -5, -6]). Checking via eig(A - B*K) confirms all four closed-loop eigenvalues land exactly at $\{-4 \pm 4i, -5, -6\}$→ the system is stable.

The gains are large — especially | k_1 |~ 10^6— because k_1 must push the open-loop integrator mode (eigenvalue at 0) all the way to -6, scaled by the small control coefficient $1/J = 0.014$.

5.4 Simulink model

The Step 4 model is reused, with the only change being an update to the Gain block. Since the block references the workspace variable K (via value -K), rerunning MTALAB code for step 5 is enough to propagate the new gains. All other blocks (plant, Mux, Signal Editor for $d(t)$, scopes) are unchanged. Initial condition stays [0; 0; 0; 0].

5.5 Simulation results with the Step 4 disturbance

Running the same trapezoidal $d(t)$ used in Step 4 (peak 0.01 m, hold $t \in [2,8]$):

Quantity	Step 4 (PID)	Step 5 (full-state)
peak ‖ θ ‖	$0.066°$	$\approx 5.84°(\approx 0.102$ rad)
peak ‖ $\dot{x}$ ‖	1.17m/s	$\approx$ 2.71m/s
$\dot{x}$ during hold	$\approx$ 0.7m/s	$\approx$ 2.52m/s
peak T	10.4N·m	$+141$and -131N·m (at corners)

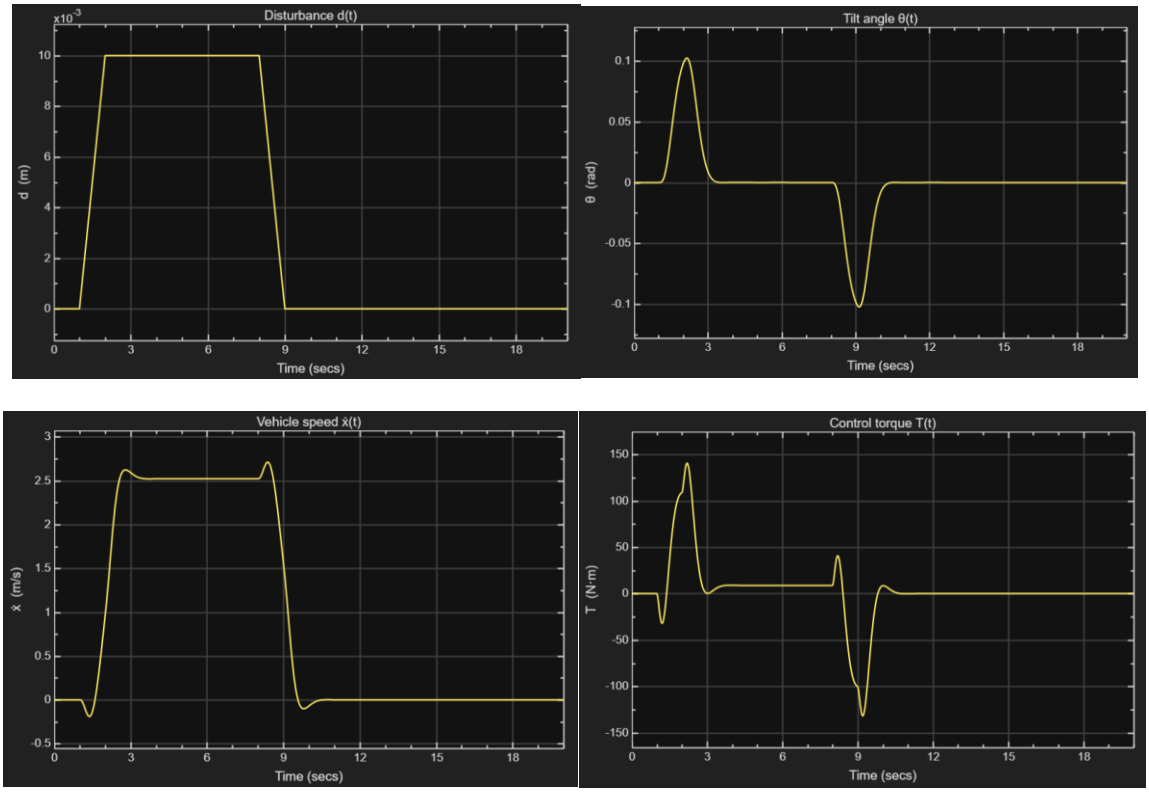


Figure 7: Scope output for disturbance, tilt angle, vehicle speed and control torque (Step 5)

5.6 Discussion

The two controllers behave quite differently for the same disturbance. In Step 5, $\dot{x}$ reaches its full static-gain asymptote (≈ 2.52 m/s) during the hold phase in under a second, because feeding back $\dot{x}$ moves the slow $c_2/(M+m) \approx -0.09$ pole to -5 (closed-loop time constant ≈ 0.2s). In Step 4, the same pole couldn't be moved (since $k_4 = 0$), so $\dot{x}$ only reached ≈ 0.7m/s during the 6-s hold.

The surprising result is that $|\theta|$ is **larger** in Step 5 (5.84° vs 0.066°). This happens because the full-state controller regulates all four states, not just θ. When $\dot{x}$ rises due to the disturbance, the controller responds by tilting the vehicle i.e., it uses θ as a means to slow $\dot{x}$. The PID controller in Step 4 only tries to drive θ to zero, which leads to small θ but slower (and incomplete) $\dot{x}$ response.

Peak torque grows by about a factor of 14 (10.4 → 141 N·m). Notably, the θ and T spikes are tightly localized at the disturbance corners; between them (e.g. $t \in [3,8]$) the states sit in a clean steady state with $\theta = 0$ and $T = 8.83$ N·m (same $mg \cdot d_{peak}$ as Step 4, since steady-state physics doesn't change).

5.7 Constraint check

Quantity	Observed peak	Constraint	Status
$\|\theta\|$	$\approx 5.84°$	$\leq 10°$	✓
$\|\dot{x}\|$	≈ 2.71m/s	$\lesssim 3$m/s	✓(tight)

Both constraints are met, though $|\dot{x}|$is close to the limit.

Step 6: Torque Saturation

Main question: What if the motor can't deliver what the controller is asking for?
Real actuators have limits, so we clip the Step 5 controller's output at 80% of its peak demand and see what happens. The answer is that the closed-loop system goes unstable through a classic integrator wind-up mechanism: while the actuator is saturated, the integrator state keeps accumulating error, eventually driving the controller into the opposite saturation limit and never recovering.

6.1 Saturation limits

From Step 5, the peak torque demand for the trapezoidal $d(t)$ is ≈ 141N·m. The saturation limit is set to 80% of this value:

$$Tmax = 0.80 \times 141 = 112.8\ N.m$$

The saturation block clips the controller's output symmetrically: $T_{\text{applied}} = \text{sat}(T_{\text{demanded}},\ \pm 112.8)$.

6.2 Simulink model

A single Saturation block is inserted in the feedback path between the Gain block output and the Mux input. The block is configured with:

- **Upper limit:** 112.8
- **Lower limit:** -112.8

Everything else (plant, Signal Editor, Mux, scopes, K, initial conditions) is identical to Step 5.

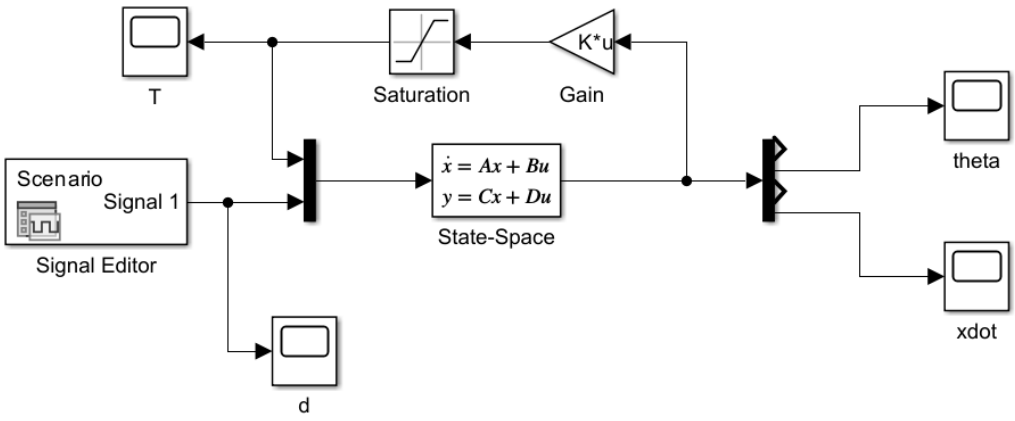


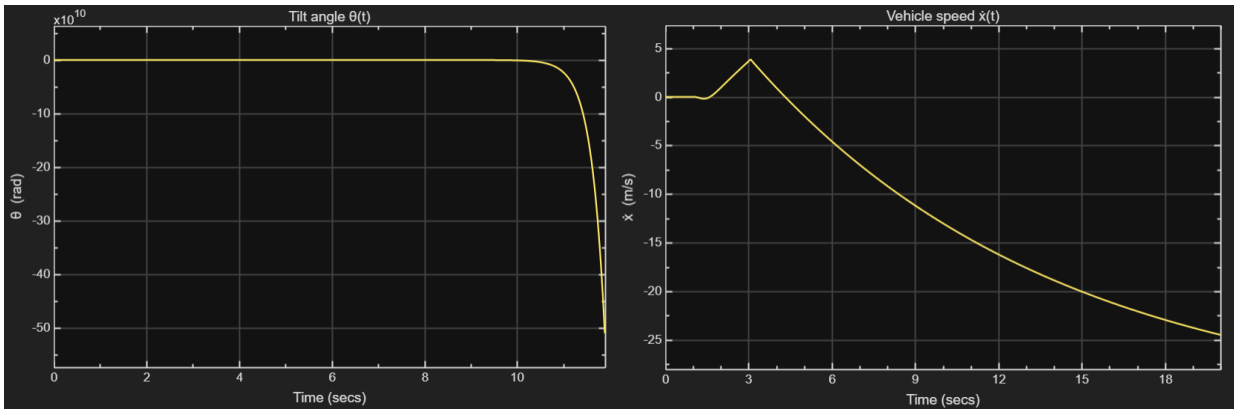
Figure 8: Simulink Model for Step 6

6.3 Simulation result

Running the same trapezoidal $d(t)$:

The closed-loop system becomes unstable. The three scope traces tell the story:

- **T(t)** — The torque starts at zero, dips briefly negative (~-30 N·m) around $t = 1.5$s, then ramps up and pins against the positive saturation limit of $+112.8$ N·m between roughly $t = 2$ and $t = 3$s. At $t \approx 3$s, the controller output flips sharply and pins against the negative limit of -112.8 N·m, where it remains for the rest of the simulation.
- **ẋ(t)** — While T is positive, the vehicle accelerates forward and ẋ peaks at $\approx +4$m/s near $t = 3$s (already above the 3 m/s target). Once T flips negative and stays there, ẋ decreases monotonically, reaching about -25m/s by $t = 20$s.
- **θ(t)** — Tilts grows slowly at first (invisible on the scope's default scale), but by the end of simulation θ has diverged to $\approx -80 \times 10^{22}$rad. The scope auto-scales to 10^{22}rad, which is the giveaway that the system has blown up.



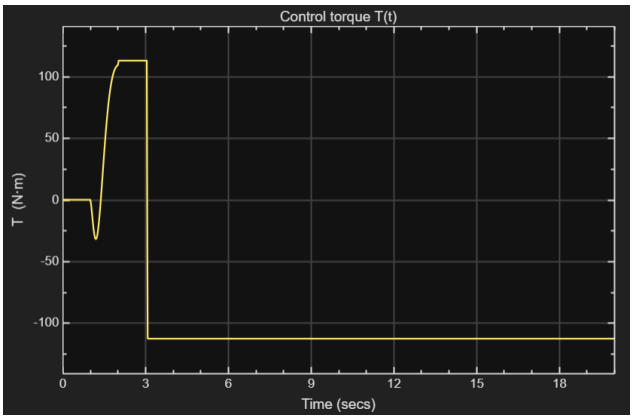


Figure 9: Scope output for tilt angle, vehicle speed and control torque (Step 6)

The positive ramp in T between $t = 2$ and $t = 3$s is the controller trying to correct the initial tilt caused by the disturbance. It saturates at $+112.8$ N·m because the demanded torque is higher than the limit. During this saturated interval, the integrator state $x_1 = \int \theta \, dt$ continues to accumulate but since the actual applied torque is insufficient, θ drifts further from zero rather than returning. By the time θ swings past equilibrium, the integrator has built up so much "error memory" that the controller now demands a huge *negative* torque which also saturates (at -112.8N·m). The plant is now receiving constant -112.8N·m forever, regardless of what the states do, and it diverges.

This is the classic **integrator wind-up** failure mode. Once the actuator saturates, the integrator's accumulated state carries the controller past the point of recovery.

6.4 Discussion

Two factors make this model particularly vulnerable to saturation-induced instability:

1. The Step 5 controller has very large gains (chosen to enforce $t_s = 1$s), so small state errors translate to torque demands far above 141 N·m if the states wander.
2. The plant is an inverted pendulum. Open-loop dynamics are destabilizing, so loss of control authority means falling over rather than just tracking poorly.

In practice, this is addressed by anti-windup schemes or by using a less aggressive controller that rarely saturates. For the purpose of this step, the takeaway is that 80% saturation is too tight for the Step 5 design with the chosen disturbance.

Step 7: Steady-State Analysis and Static Gain

Main question: Can we predict the steady-state speed for a given lean, without simulating?
For a constant disturbance, the closed-loop dynamics settle to a fixed point that can be computed directly from the state-space equations. We solve for x_ss, extract the static gain from d to ẋ (252.26 s^{-1} for the Step 5 controller), and verify against a Simulink step response. Matching the two confirms the model, the controller, and the simulation are all consistent.

7.1 Steady-state equations

The closed-loop dynamics are

$$\dot{x} = (A - BK)\, x + F\, d$$

For a constant (step) disturbance $d(t) = d_{ss}$, steady state requires $\dot{x} = 0$:

$$0 = (A - BK)\, x_{ss} + F\, d_{ss} \Longrightarrow x_{ss} = -(A - BK)^{-1}\, F\, d_{ss}$$

This inverse exists because the eigenvalues of $A - BK$ are all in the LHP (Step 5), so $A - BK$ is non-singular.

7.2 Computation for unit step in d

Setting $d_{ss} = 1$m and solving $x_{ss} = -(A - BK)^{-1}F$:

$$x_{ss} = \begin{bmatrix} \int \theta\, dt \\ \theta \\ \dot{\theta} \\ \dot{x} \end{bmatrix}_{ss} = \begin{bmatrix} 10.70 \\ \approx 0 \\ 0 \\ 252.26 \end{bmatrix}$$

*In MATLAB, this is computed as x_ss = -(A - B*K) \ F (using the backslash operator).*

The static gain from d to $\dot{x}$ is therefore:

$$\boxed{\frac{\dot{x}_{ss}}{d_{ss}} = 252.26\ s^{-1}}$$

Two observations from x_{ss}:

- $\theta_{ss} = 0$. The integral action ($k_1 \int \theta\, dt$) drives the tilt to zero at steady state for any constant disturbance. This is the defining behavior of a PID / integral-action controller.
- $\left(\int \theta\, dt\right)_{ss} = 10.70$. This state accumulates to a non-zero value because the controller needs it to generate the steady-state torque. Specifically, $T_{ss} = -Kx_{ss} = 882.9$ N·m, which equals $mg\, d_{ss} = 882.9$ which is exactly the torque needed to cancel the gravitational disturbance torque.

7.3 Comparison with the open-loop static gain (Step 4)

In Step 4, the same static gain $\dot{x}_{ss}/d_{ss} = mg/(Rc_2) = 252.26$ was derived from the open-loop plant. The closed-loop static gain matches exactly. This makes sense physically: at steady state, the controller perfectly regulates θ back to zero, so the only remaining dynamics are the ẋ equation (II), and its steady-state balance doesn't depend on the controller.

7.4 Simulink verification

Because a unit step in d ($d = 1$ m) would predict $\dot{x}_{ss} = 252$m/s, a physically meaningless value far outside the operating range, the verification uses a small step $d_{ss} = 0.001$m, which gives the scaled prediction

$$\dot{x}_{ss}^{\text{predicted}} = 252.26 \times 0.001 = 0.2523 \text{ m/s}$$

Simulink setup. Starting from the Step 5 model (no saturation):

- Replace the Signal Editor block with a Step block (Simulink → Sources).
- Configure the Step block: Step time = 1, Initial value = 0, Final value = 0.001.
- Wire to the disturbance channel of the Mux.
- Initial conditions: [0; 0; 0; 0].
- Stop time: 5 s (enough to reach steady state, given the slowest closed-loop pole at -5 gives a settling time of about 1 s).

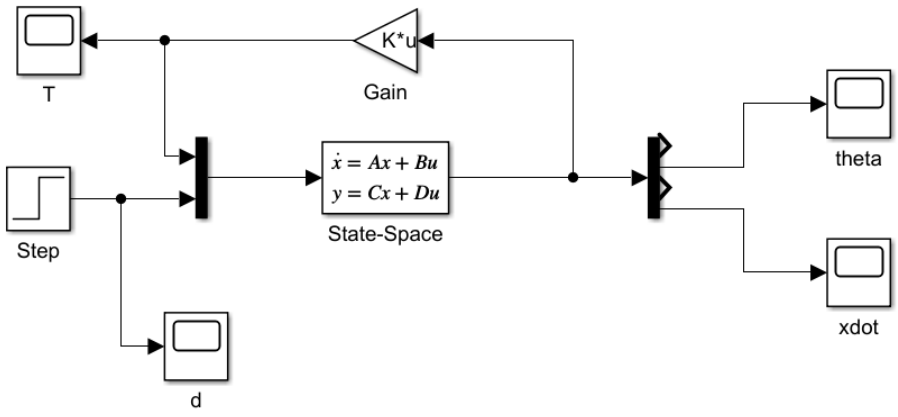
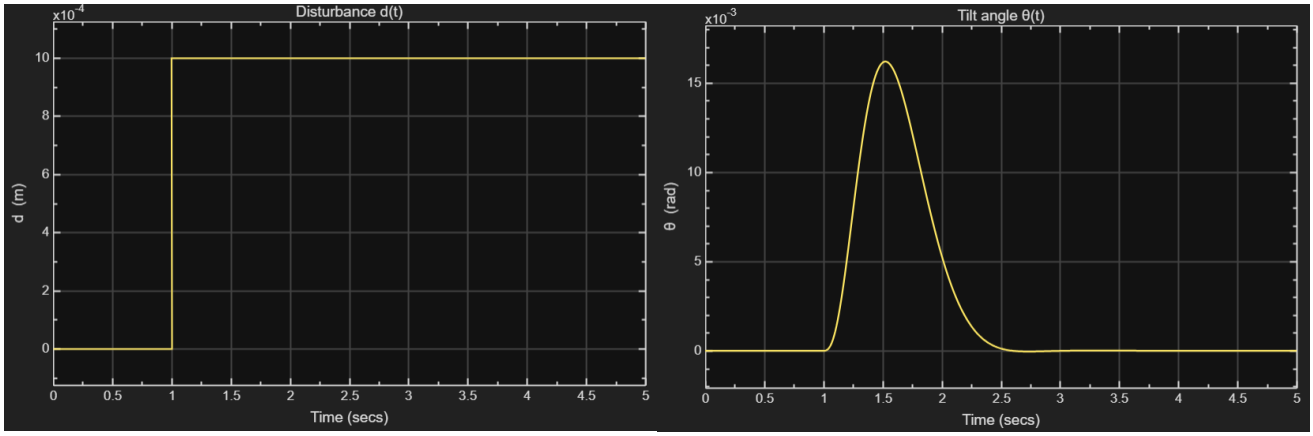


Figure 10: Simulink Model for Step 7

Expected vs. observed:

Quantity	Predicted	Observed (from scope)
$\dot{x}_{ss}$	0.2523m/s	$\approx$ 0.25m/s
θ_{ss}	0	$\approx$ 0
T_{ss}	0.883N·m	$\approx$ 0.88N·m

The simulation confirms the analytical prediction. Multiplying the observed steady-state speed by $1/d_{ss} = 1000$ recovers a static gain of approximately 250s^{-1}, matching the computed value of 252.26s^{-1} to within the precision of reading off the scope.



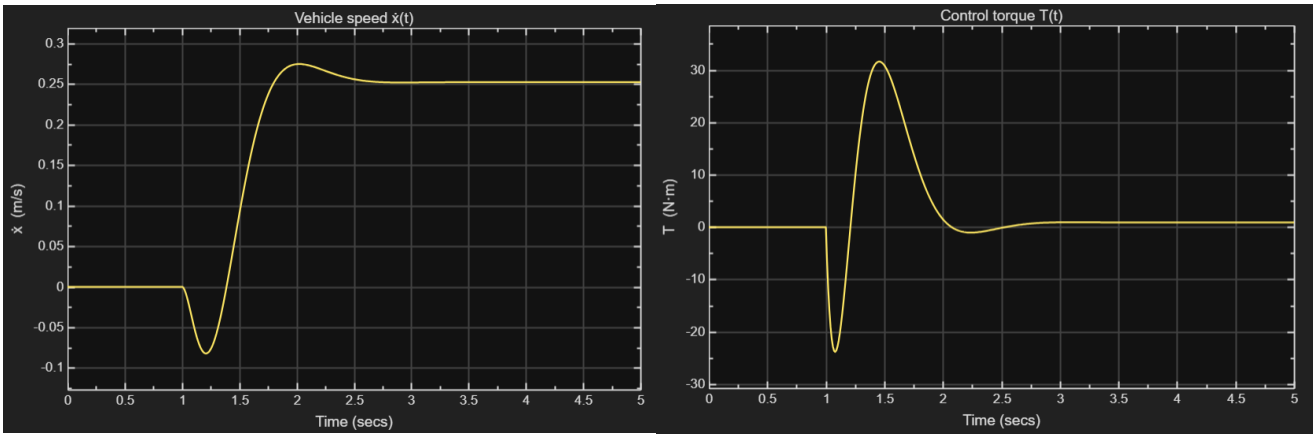


Figure 11: Scope output for disturbance, tilt angle, vehicle speed and control torque (Step 7)

Transient behavior for reference:

- $\theta(t)$ peaks at ≈ 0.016rad ($\approx 0.92°$) near $t = 1.5$s before settling to zero.
- $\dot{x}(t)$ first dips to about -0.08m/s (the controller initially tilts the vehicle backward before recovering), overshoots to ≈ 0.275m/s at $t \approx 2$s, and settles to 0.25m/s.
- $T(t)$ swings between -24 and $+32$N·m during the transient before settling to ≈ 0.88N·m.

Step 8: Reduced 3-State Model with Full-State Feedback

Main question: What do we lose (or gain) by dropping the integrator?
We repeat the Step 5 design with only three states (θ, θ̇, ẋ), no ∫θ. Without the integrator, θ no longer returns to zero under a constant disturbance. The vehicle settles at a small backward tilt. But that tilt does real work: it lets gravity cancel most of the disturbance torque, so the motor barely has to supply anything. The result is much smaller torques, speeds, and power at the cost of a permanent (small) θ offset.

8.1 New state definition

The state vector now drops the integrator term:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \dot{x} \end{bmatrix}$$

All three states are assumed measurable.

8.2 State-space matrices

Re-deriving directly from the EoMs with the new state ordering:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{MgL + mg\ell}{J} & -\dfrac{c_1}{J} & 0 \\ 0 & 0 & -\dfrac{c_2}{M+m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -\dfrac{1}{J} \\ \dfrac{1}{R(M+m)} \end{bmatrix}, \quad F = \begin{bmatrix} 0 \\ \dfrac{mg}{J} \\ 0 \end{bmatrix}$$

With numerical values:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 12.0770 & -0.01407 & 0 \\ 0 & 0 & -0.09091 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -0.01407 \\ 0.02597 \end{bmatrix}, \quad F = \begin{bmatrix} 0 \\ 12.4221 \\ 0 \end{bmatrix}$$

8.3 Controllability

The Kalman controllability matrix is

$$\mathcal{C} = [B,\ AB,\ A^2B] = \begin{bmatrix} 0 & -0.01407 & 0.000198 \\ -0.01407 & 0.000198 & -0.1699 \\ 0.02597 & -0.00236 & 0.000215 \end{bmatrix}$$

Numerical rank = 3, so (A, B) is fully controllable.

In MATLAB, this is computed with ctrb(A, B) and rank(ctrb(A, B)).

8.4 Pole placement

Using the same settling-time spec as Step 5 ($t_s \approx 1$ s $\Rightarrow \sigma = 4$):

- Dominant pair: $-4 \pm 4j$
- Non-dominant: -5

Pole placement via MATLAB place yields:

$$K = [k_1 \quad k_2 \quad k_3] \approx [-5822.9 \quad -1820.8 \quad -489.8]$$

In MATLAB: K = place(A, B, [-4+4i, -4-4i, -5]). Verification via eig(A - B*K) confirms the closed-loop eigenvalues land exactly at $\{-4 \pm 4j, -5\}$ — all LHP → stable. ✓

Note on gain magnitudes. The gains here are about 100× smaller than Step 5 (where $| k_1 | \sim 10^6$). The reason is the absence of the integrator state $x_1 = \int \theta\, dt$. In Step 5, k_1 had to scale with the product of four pole magnitudes whereas here it scales with only three, and the integrator mode itself is gone from the plant.

8.5 Simulink model

The plant structure is the same (State-Space block with input $[\mathrm{T};\ \mathrm{d}]$, Mux, Signal Editor for $d(t)$), but the plant block is resized to 3 states:

- A: the 3×3 matrix from Step 8.2
- B: [B F] (3×2)
- C: eye(3)
- D: zeros(3,2)
- Initial conditions: [0; 0; 0]

The Gain block uses the new 1×3 gain vector -K. All other wiring is unchanged from Step 5.

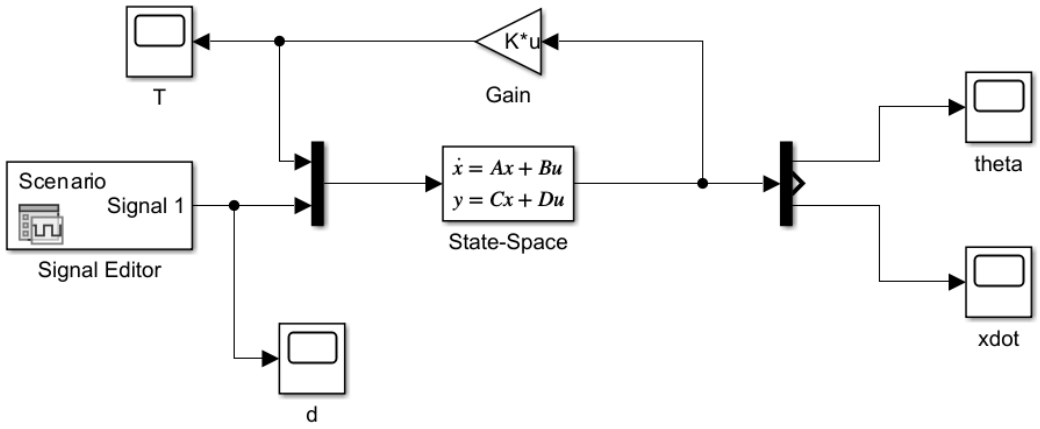
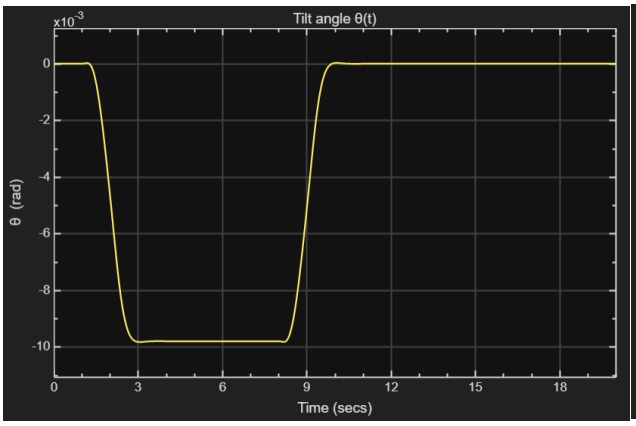


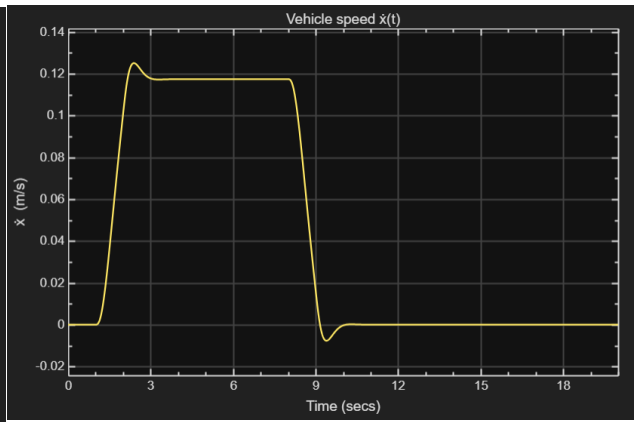
Figure 12: Simulink Model for Step 8

8.6 Simulation results with the Step 4 disturbance

Running the same trapezoidal $d(t)$(peak 0.01 m, hold $t \in [2,8]$):

Quantity	Step 5 (4-state)	Step 8 (3-state)
peak ‖ θ ‖	5.84°	≈ 0.56°
peak ‖ $\dot{x}$ ‖	2.71m/s	≈ 0.125m/s
peak ‖ T ‖	141N·m	≈ 5.4N·m (positive), ≈ −4.7N·m (negative)
θ during hold	0	≈ −0.56°(steady-state error)
$\dot{x}$ during hold	2.52m/s	≈ 0.118m/s
T during hold	8.83N·m	≈ 0.5N·m





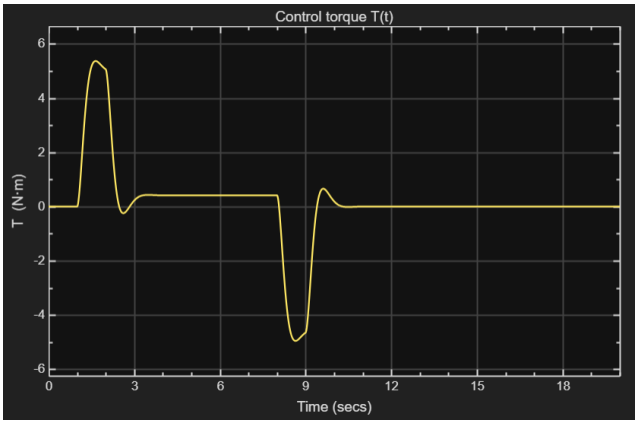


Figure 13: Scope output for tilt angle, vehicle speed and control torque (Step 8)

8.7 Discussion

The 3-state controller is much less aggressive than Step 5 — peak θ, peak ẋ, and peak torque are all roughly an order of magnitude smaller. Two things to note:

Nonzero steady-state θ. Without the integrator state, there is no mechanism to drive θ_{ss} to zero when a constant disturbance is present. During the hold phase, θ settles at about $-0.56°$ instead of zero. Physically, the vehicle leans slightly backward while the rider holds their forward lean — a permanent offset in response to a permanent disturbance. Step 5's integrator removed this offset; Step 8 doesn't have it, so the offset stays.

Much smaller $\dot{x}$ during hold. The static gain from d to $\dot{x}$ drops from 252.26s^{-1} (Step 5) to 11.74s^{-1} (Step 8), a factor of ~20 smaller. The reason: with the feedback gain k_3 acting directly on $\dot{x}$, the closed-loop "effective damping" on the horizontal dynamics is much larger than the physical c_2, so a constant disturbance produces much less steady-state velocity. Step 5's integrator forced $\theta_{ss} = 0$, which *removed* the θ-based damping mechanism on ẋ and let ẋ reach its unimpeded static gain.

8.8 Constraint check

Quantity	Observed peak	Constraint	Status
$\| \theta \|$	$\approx 0.56°$	$\leq 10°$	✓
$\| \dot{x} \|$	$\approx 0.13\text{m/s}$	$\lesssim 3\text{m/s}$	✓

Both met with large margin. The 3-state controller is conservative for this disturbance.

Step 9: Peak Torque and Peak Power

Main question: How much work is each controller actually doing?
Running all three designs (Steps 4, 5, 8) under the same disturbance, we read off peak torque and compute peak mechanical power P = T·ẋ/R. The three controllers span three orders of magnitude in power of about 29 W, 717 W, and 1.5 W respectively. The ranking is a direct reflection of each design's philosophy: Step 8 lets posture do the work, Step 5 forces the motor to do everything, and Step 4 sits in between.

All three simulations (Steps 4, 5, 8) are run under the same condition: the trapezoidal disturbance $d(t)$ from Step 4, with $d_{peak} = 0.01$m, ramps at $t \in [1,2]$ and $[8, 9]$, hold on $t \in [2,8]$. This way the only thing that varies between simulations is the controller.

9.1 Definition of power

The motor torque T acts at the wheel axis. The wheel's angular velocity is $\omega_{\text{wheel}} = \dot{x}/R$ (since the wheel rolls without slipping). The mechanical power delivered to the Segway is therefore

$$P(t) = T(t)\,\omega_{\text{wheel}}(t) = \frac{T(t)\,\dot{x}(t)}{R}$$

With $R = 0.35$m, the conversion factor is $1/R \approx 2.857$ rad/m.

Power was computed inside Simulink by adding a Product block that multiplies the torque signal T by the speed signal $\dot{x}$, followed by a Gain block with value $1/R$ that divides by the wheel radius. The result is routed to a dedicated Scope labeled $P(t)$. This branch was added to each of the three Simulink models so that peak power could be read directly off the scope after running each simulation. Showing the Simulink of Step 8. (ẋ is output 4 for 4-state models, output 3 for the Step 8 3-state model).

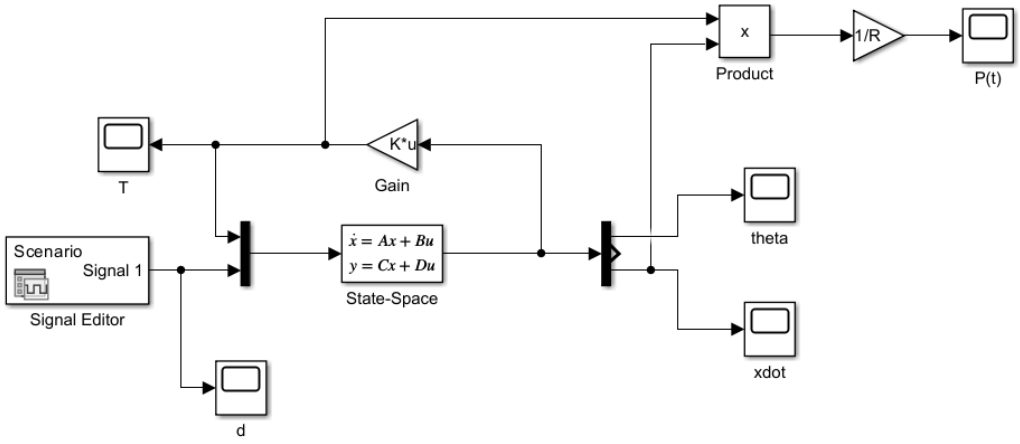


Figure 14: Simulink Model for Step 9

9.2 Results

Step	Controller	peak ‖ T ‖(N·m)	peak ‖ $\dot{x}$ ‖(m/s)	peak ‖ P ‖(W)
4	PID, 4-state ($k_4 = 0$)	10.4	1.17	≈ 29
5	Full-state, 4-state	140.6	2.71	≈ 717
8	Full-state, 3-state	5.37	0.125	≈ 1.5

The peak power values are read directly from the $P(t)$ Scope in each model, using the Data Cursor tool to pick off the peak magnitude.

Observed shapes from the power scopes:

- **Step 4:** $P(t)$ rises steadily as $\dot{x}$ builds up during the hold (since T is roughly constant), peaking at about 29W near $t \approx 8$s, then swings briefly negative (~-5 W) as the disturbance ramps down.
- **Step 5:** A sharp peak at $+717$W near $t \approx 2$s, a symmetric dip to about -540W near $t \approx 9$s, and a flat hold segment at approximately $+70$W (consistent with $T_{ss} \cdot \dot{x}_{ss}/R = 8.83 \cdot 2.52/0.35 \approx 63.5$W, small difference due to transient overshoot during the hold).
- **Step 8:** Small peak of $+1.48$W near $t \approx 2$s, symmetric dip of -1.1W near $t \approx 9$s, nearly zero during the hold since both $T$and $\dot{x}$are small there.

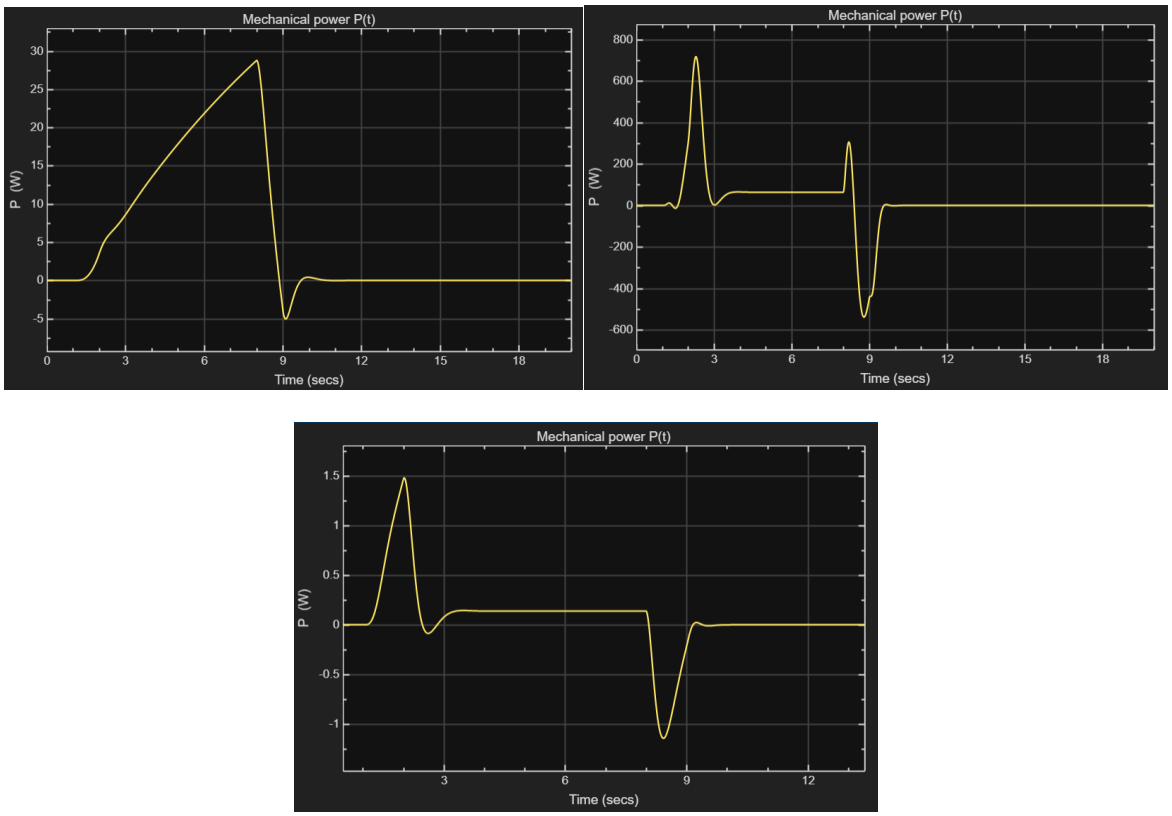


Figure 15: Scope output of P(t) for Step 4, Step 5 and Step 8 models

9.3 Discussion

The three controllers span three orders of magnitude in power demand:

- **Step 8 (3-state)** is the most efficient with peak power under 2 W. The controller barely moves the vehicle (ẋ stays under 0.13 m/s), and the torque never exceeds 5.4 N·m.
- **Step 4 (PID)** sits in the middle at ~29 W. The large gains needed to beat gravity give modest torque peaks, and the vehicle does accelerate but only to ~1.17 m/s.
- **Step 5 (full-state)** dominates at ~717 W. The controller drives ẋ to its full static-gain asymptote of 2.52 m/s and applies ~140 N·m at the disturbance corners.

The ranking in peak power mirrors the aggression of each design: Step 8 has no integrator and small gains → small torque and small ẋ → small power. Step 5 has an integrator *and* high-bandwidth placement of the ẋ pole → large torque *and* large ẋ → power scales as their product. Step 4 is in between because it has the integrator but can't regulate ẋ.

This is consistent with the observation from Step 6: Step 5's 141 N·m demand is what gets clipped by a saturation limit (and triggers wind-up), while the Step 4 and Step 8 designs would tolerate even tight saturation easily.

Step 10: Full-State Observer and Controller with Estimated Speed

Main question: What if we can't measure every state directly?
In practice, some states have to be estimated rather than measured. An observer is a mathematical copy of the plant that runs inside the controller, uses whatever measurements are available, and corrects itself using the prediction error. We design observers for both the 4-state and 3-state models with poles about 5× faster than the controller, then close the loop using the estimated speed in place of the true one. Aside from a brief transient when the observer is deliberately initialized with a wrong guess, the system behaves exactly as it did with true-state feedback.

10.1 Observability from θ alone

If only θ is measured, the output matrix is

$$C = [0 \quad 1 \quad 0 \quad 0] \text{ (4-state) or } C = [1 \quad 0 \quad 0] \text{ (3-state)}$$

The Kalman observability matrix $\mathcal{O} = [C;\ CA;\ CA^2;\ \ldots]$ has **rank 2** in both cases so the system is **not fully observable** from θ alone.

In MATLAB, this is checked with rank(obsv(A, C)). For the 4-state case with C = [0 1 0 0], the returned rank is 2, indicating 2 unobservable modes.

This makes physical sense: Eq. (II), $(M+m)\ddot{x} = T/R - c_2\dot{x}$, does not contain θ, so measuring θ gives no information about x or $\dot{x}$.

10.2 Using $C = I$ for the observer

To work around the observability limitation, the observer is built with $C = I$ (identity). With full-state "measurement," the system is trivially observable ($\mathrm{rank}(\mathcal{O}) = n$), and a full-state observer can be designed. The estimated speed $\hat{x}$ produced by this observer is then used in place of the true $\dot{x}$ in the controller, while the other states are taken as measured.

10.3 Observer design

The observer has dynamics

$$\dot{\hat{x}} = A\,\hat{x} + B\,T + L\,(y - C\,\hat{x}) \text{ with } C = I$$

so the estimation error $e = x - \hat{x}$ evolves according to

$$\dot{e} = (A - LC)\,e = (A - L)\,e$$

For fast convergence, the observer eigenvalues are placed 3–10× faster than the controller eigenvalues.

Using the controller dominant real part of 4 from Steps 5 and 8:

- **4-state observer poles:** $\{-20 \pm 20j,\ -25,\ -30\}$(5× faster than controller)
- **3-state observer poles:** $\{-20 \pm 20j,\ -25\}$

In MATLAB, the observer gain is computed via duality: L = place(A', eye(n), obs_poles).'.

Resulting gains (from MATLAB's place):

$$L_{\text{4-state}} = \begin{bmatrix} 20 & -19 & 0 & 0 \\ 20 & 20 & 1 & 0 \\ 0 & 12.08 & 24.99 & 0 \\ 0 & 0 & 0 & 29.91 \end{bmatrix}, \; L_{\text{3-state}} = \begin{bmatrix} 20 & -19 & 0 \\ 32.08 & 19.99 & 0 \\ 0 & 0 & 24.91 \end{bmatrix}$$

Verifying with eig(A - L*C) confirms the eigenvalues land at the specified observer poles:

- 4-state: $\{-20 \pm 20j, \; -25, \; -30\}$✓
- 3-state: $\{-20 \pm 20j, \; -25\}$✓

10.4 Observer convergence (standalone check)

With a non-zero initial estimation error $\hat{x}(0) - x(0) = [0,\ 0,\ 0,\ 0.05]^T$ for the 4-state case (estimated $\dot{x}$ starts at 0.05 m/s while the true $\dot{x}(0) = 0$):

- At $t = 0.5$s, error decays to $\sim 10^{-6}$m/s.
- By $t = 1$s, error is at numerical noise level.

Observer settling time is roughly 0.2 s, consistent with the dominant observer pole at -20.

10.5 Controller with mixed feedback

In the closed-loop test, the controller uses:

- **Measured values** for $\int \theta \, dt$, θ, $\dot{\theta}$
- **Estimated value** $\hat{x}$ from the observer in place of the true $\dot{x}$

i.e. $T = -K\, x_{\text{mix}}$ where $x_{\text{mix}} = [\int \theta \, dt,\ \theta,\ \dot{\theta},\ \hat{x}]^T$ for the 4-state case.

10.6 Simulation results

Running the same trapezoidal $d(t)$ as Steps 4–8, with initial state $x(0) = [0,\ 0,\ 0,\ 0]$ and a deliberate initial mismatch $\hat{x}(0) = [0,\ 0,\ 0,\ 0.05]$:

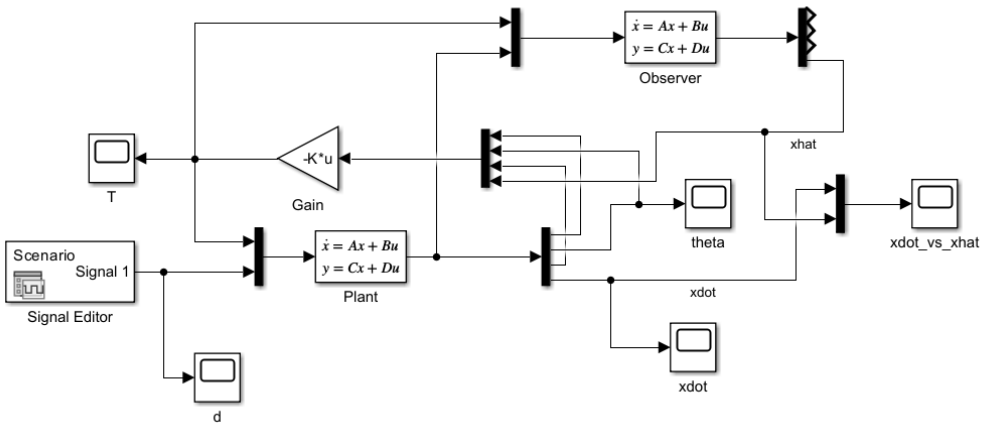


Figure 16: Simulink Model for 4-state case Step 10

4-state case (Step 5 controller + observer):

Quantity	Step 5 (true states)	Step 10 (estimated $\dot{x}$)
peak $\| \theta \|$	$5.84°$	$\approx 5.84°$(post-transient, matches Step 5 exactly)
peak $\| \dot{x} \|$	2.71m/s	≈ 2.71m/s (post-transient); brief initial dip to ≈ -0.55m/s
peak $\| T \|$	141N·m	≈ -1600N·m at $t = 0$(observer-transient spike); ≈ 150N·m at the corners (like Step 5)

Observed traces:

- $\theta(t)$: brief oscillation of magnitude ≈ 0.025rad ($\approx 1.4°$) during $t \in [0,0.3]$s while the observer converges, then settles and produces the Step-5 shape (+0.102 rad at $t \approx 2$s, -0.102rad at $t \approx 9$s).
- $\dot{x}(t)$: initial dip to ≈ -0.55m/s due to the initial estimation mismatch, then rises to the expected hold value of ≈ 2.52m/s, with corner overshoots at ± 2.71m/s as in Step 5.
- $T(t)$: dominant feature is the downward spike to ≈ -1600N·m at $t = 0$, arising from the controller applying $-k_4 \cdot \hat{\dot{x}}(0) = -31838 \cdot 0.05 \approx -1592$N·m. After observer convergence, T settles into the Step-5 pattern.

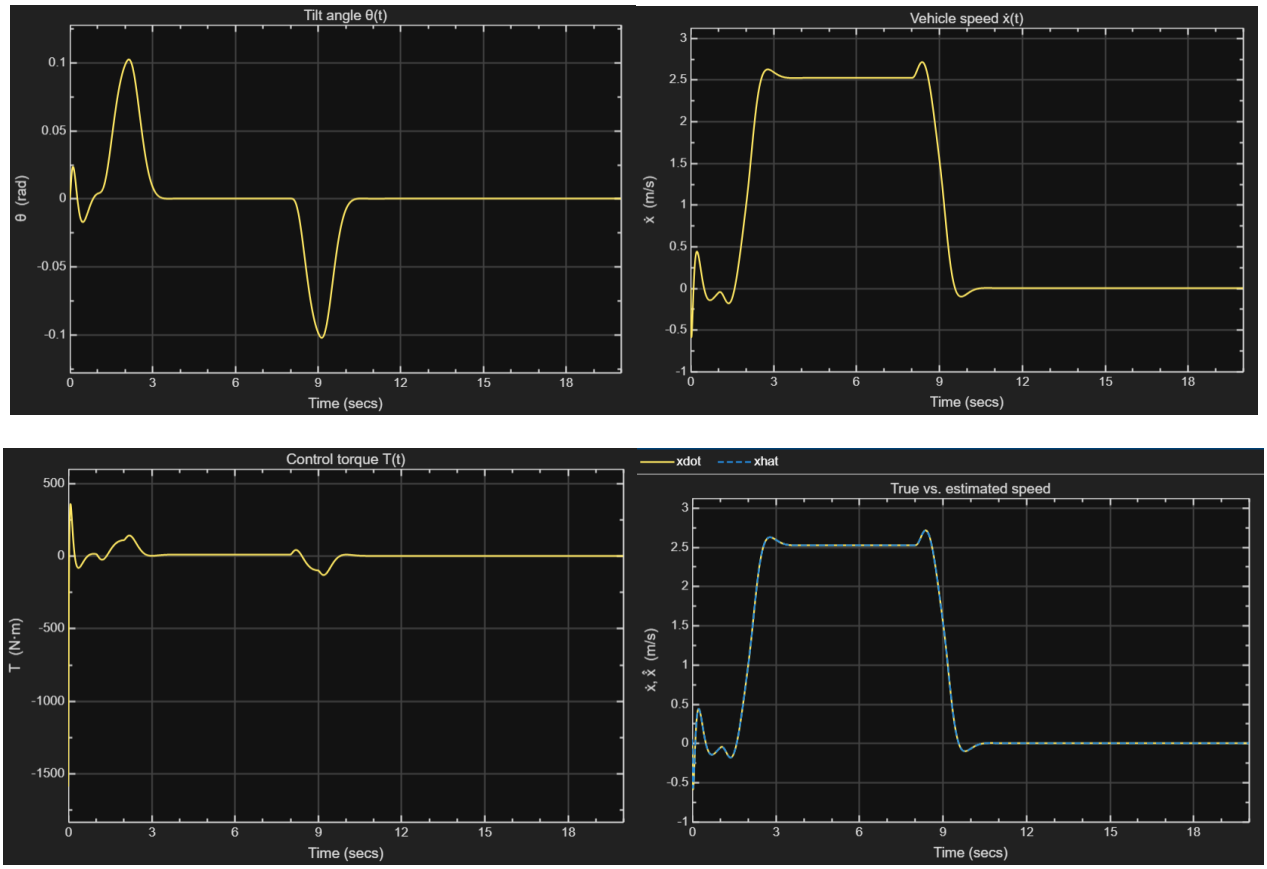


Figure 17: Scope output for tilt angle, vehicle speed, control torque and true vs estimated speed (4-state case 10 Step)

3-state case (Step 8 controller + observer):

the same construction was repeated with the 3-state model and Step 8 controller, using the 3-state observer gain $L_{\text{3-state}}$and initial estimation error $\hat{x}(0) - x(0) = [0,\ 0,\ 0.05]^T$.

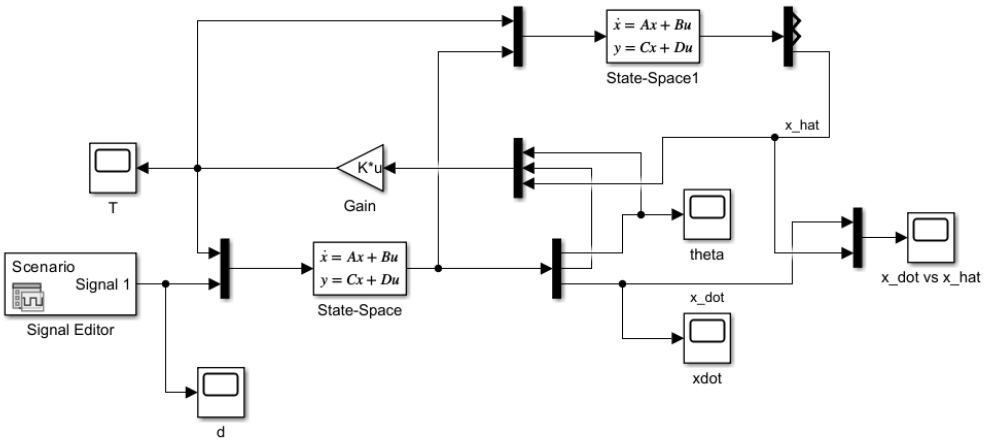
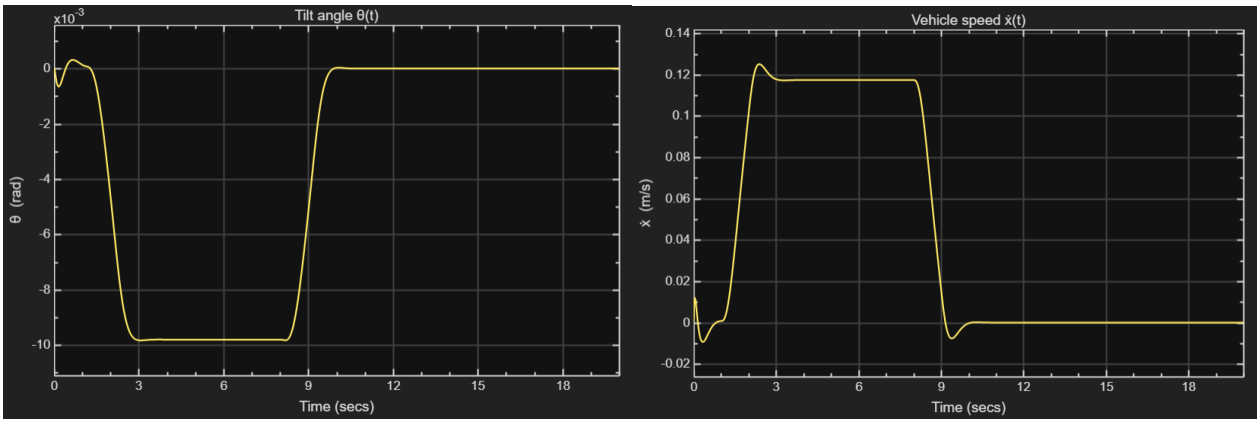


Figure 18: Simulink Model for 3-state case Step 10

Quantity	Step 8 (true states)	Step 10 (estimated $\dot{x}$)
peak ‖ θ ‖	$0.56°$	$\approx 0.56°$(post-transient, matches Step 8); brief initial bump of $\sim 0.03°$
peak ‖ $\dot{x}$ ‖	0.125m/s	≈ 0.125m/s (post-transient, matches Step 8)
peak ‖ T ‖	5.4N·m	≈ -5N·m at $t = 0$(observer-transient spike); ≈ 5.4N·m at the corners (like Step 8)

The initial torque spike for the 3-state case is much smaller than the 4-state spike (−5 vs −1600 N·m) because the controller's velocity gain | k_3 |≈ 490 is about 65× smaller than Step 5's | k_4 |≈ 31838. Both observers converge essentially instantaneously (within ~0.1 s) due to the fast observer poles.

The "true vs. estimated speed" comparison scope shows the yellow $\dot{x}$ trace and dashed cyan $\hat{\dot{x}}$ trace overlapping for nearly the entire simulation, with a barely-visible initial separation at $t = 0$ that closes within the first 0.1 second confirming the separation principle in action.



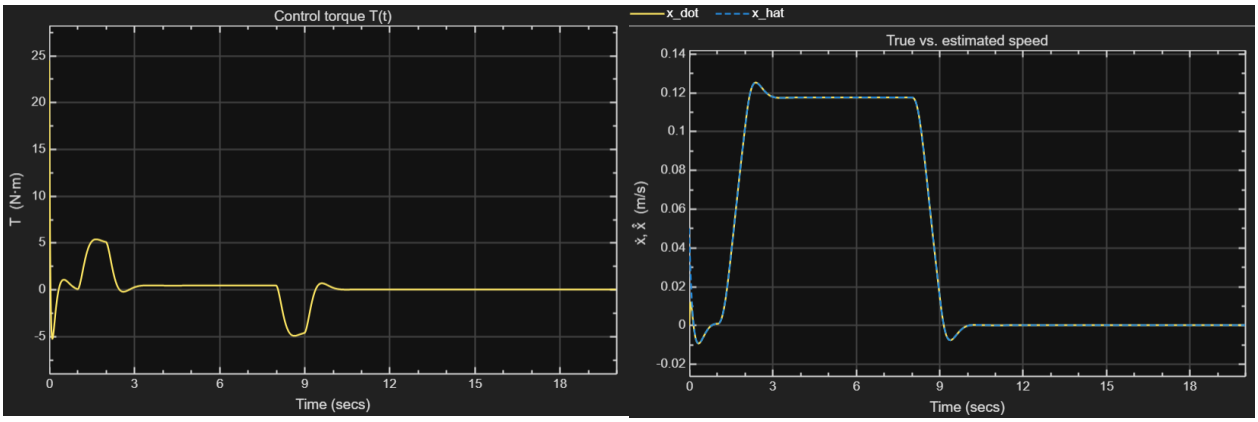


Figure 19: Scope output for tilt angle, vehicle speed, control torque and true vs estimated speed (3-state case 10 Step)

10.7 Discussion

The system still behaves as expected. Peak θ and peak ẋ are essentially unchanged but a torque spike appears during the observer's initial convergence window.

Why the spike. At $t = 0$, the estimate $\hat{x}(0) = 0.05$m/s disagrees with the true $\dot{x}(0) = 0$. The controller, fed this wrong estimate, applies a torque proportional to $-k_4 \cdot \hat{x}(0) = -31838 \cdot 0.05 \approx -1592$N·m closely matching the observed initial downward spike of about -1600N·m. Once the observer converges (within about 0.3 s), the estimate matches the truth and normal operation resumes. The brief θ oscillation (~1.4°) and $\dot{x}$ dip (~−0.55 m/s) in the same 0.3-second window are the plant's response to this initial incorrect torque.

Separation principle holds. After the observer transient, θ and $\dot{x}$ peaks match the true-state case to 4 digits, because the closed-loop eigenvalues factor into the controller and observer sets independently.

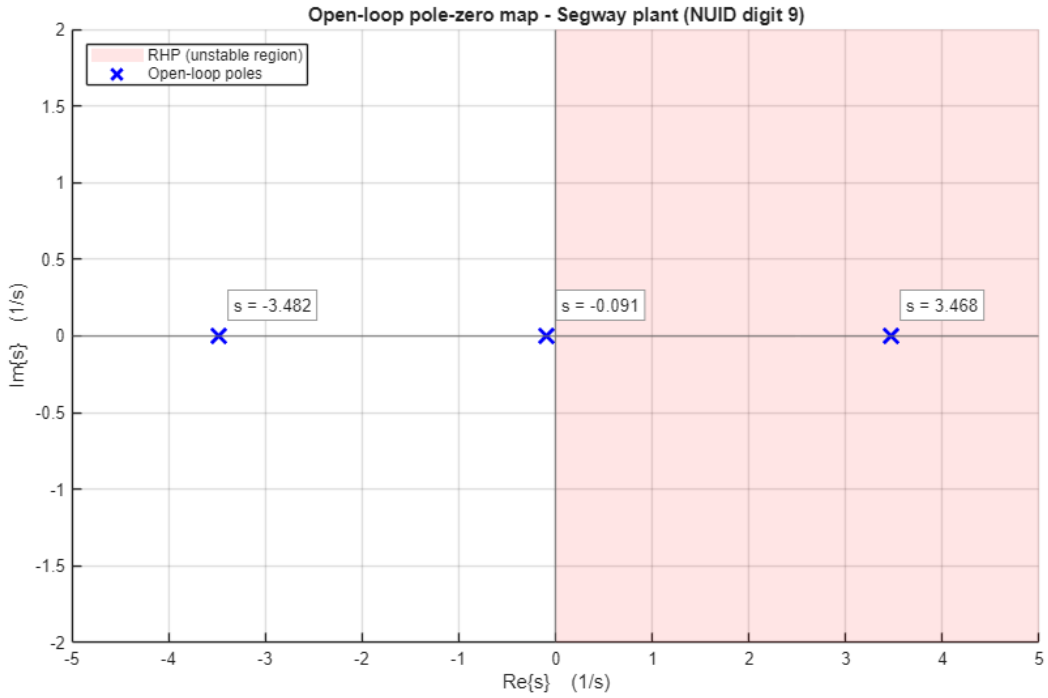
Practical implication. Initializing the observer to match the expected initial state ($\hat{x}(0) = x(0)$) would eliminate the spike entirely. The spike shown here is a deliberate test of observer behavior under mismatch. The spike magnitude scales with the corresponding velocity gain in the controller: the 3-state case showed a spike of only about -5N·m (vs -1600N·m for the 4-state case), a ratio that lines up with the gain ratio $|k_4|/|k_3| \approx 31838/490 \approx 65$.

Note on observability limitation. This demonstration uses $C = I$ (full-state measurement), which sidesteps the fact that the true system is not observable from θ alone. In a real Segway, an encoder on the wheels would be added to measure $\dot{x}$ directly, making the system fully observable with physical sensors.

```
% ME 5659 Term Project - Step 1

clear; clc; close all;

% Parameters
R   = 0.35;
L   = 0.55;
ell = 0.85;
M   = 20;
m   = 90;
c1  = 1;
c2  = 10;
g   = 9.81;

J = M*L^2 + m*ell^2;

% Transfer functions
s = tf('s');

G_theta_T = -1 / (J*s^2 + c1*s - (M*g*L + m*g*ell));
G_theta_D = (m*g) / (J*s^2 + c1*s - (M*g*L + m*g*ell));
G_xdot_T  =  1  / (R*((M+m)*s + c2));
G_xdot_D  =  0;

disp('Theta(s)/T(s):');  G_theta_T
disp('Theta(s)/D(s):');  G_theta_D
disp('Xdot(s)/T(s):');   G_xdot_T
disp('Xdot(s)/D(s) = 0');

% Poles
poles_theta = pole(G_theta_T);
poles_xdot  = pole(G_xdot_T);

disp('Poles of theta subsystem:'); disp(poles_theta);
disp('Poles of xdot subsystem:');  disp(poles_xdot);

% Stability
all_poles = [poles_theta; poles_xdot];
if any(real(all_poles) > 0)
    disp('System is UNSTABLE (RHP pole present).');
else
    disp('System is stable.');
end

% Pole-zero map
figure;
% Shade the right-half plane
patch([0 5 5 0], [-2 -2 2 2], 'r', 'FaceAlpha', 0.1, 'EdgeColor', 'none');
hold on;
% Axes through origin
xline(0, '-', 'Color', [0.4 0.4 0.4]);
yline(0, '-', 'Color', [0.4 0.4 0.4]);
```

```
% Pole markers
plot(real(all_poles), imag(all_poles), 'bx', 'MarkerSize', 12, 'LineWidth',
2);
% Label each pole
for k = 1:length(all_poles)
    text(real(all_poles(k))+0.15, imag(all_poles(k))+0.2, ...
        sprintf('s = %.3f', all_poles(k)), 'FontSize', 10, ...
        'BackgroundColor', 'w', 'EdgeColor', [0.6 0.6 0.6]);
end
grid on;
xlim([-5 5]);
ylim([-2 2]);
xlabel('Re\{s\}  (1/s)');
ylabel('Im\{s\}  (1/s)');
title('Open-loop pole-zero map - Segway plant (NUID digit 9)');
legend({'RHP (unstable region)', '', '', 'Open-loop poles'}, 'Location',
'northwest');
```

```
Theta(s)/T(s):

G_theta_T =

          -1
  ---------------------
  71.07 s^2 + s - 858.4

Continuous-time transfer function.
Theta(s)/D(s):

G_theta_D =

         882.9
  ---------------------
  71.07 s^2 + s - 858.4

Continuous-time transfer function.
Xdot(s)/T(s):

G_xdot_T =

       1
  ------------
  38.5 s + 3.5

Continuous-time transfer function.
Xdot(s)/D(s) = 0
Poles of theta subsystem:
   -3.4822
    3.4682

Poles of xdot subsystem:
   -0.0909

System is UNSTABLE (RHP pole present).
```



Published with MATLAB® R2025a

```
% Step 2
% State-space form, controllability, output controllability

clear; clc;

% Parameters
R   = 0.35;
L   = 0.55;
ell = 0.85;
M   = 20;
m   = 90;
c1  = 1;
c2  = 10;
g   = 9.81;

J = M*L^2 + m*ell^2;

% State-space matrices
A = [0, 1,                    0,        0;
     0, 0,                    1,        0;
     0, (M*g*L + m*g*ell)/J, -c1/J,     0;
     0, 0,                    0,        -c2/(M+m)];

B = [0;
     0;
     -1/J;
     1/(R*(M+m))];

F = [0;
     0;
     m*g/J;
     0];

C = [0 1 0 0;
     0 0 0 1];

disp('A ='); disp(A);
disp('B ='); disp(B);
disp('F ='); disp(F);
disp('C ='); disp(C);

% State controllability
Cctrb = ctrb(A, B);
disp('Controllability matrix ='); disp(Cctrb);
fprintf('rank(ctrb) = %d  (need 4 for full controllability)\n\n',
rank(Cctrb));

% Output controllability
Coc = C * Cctrb;
disp('Output controllability matrix ='); disp(Coc);
fprintf('rank(output ctrb) = %d  (need 2 for output controllability)\n',
rank(Coc));
```

```
A =
         0    1.0000         0         0
         0         0    1.0000         0
         0   12.0770   -0.0141         0
         0         0         0   -0.0909

B =
         0
         0
   -0.0141
    0.0260

F =
         0
         0
   12.4221
         0

C =
     0     1     0     0
     0     0     0     1

Controllability matrix =
         0         0   -0.0141    0.0002
         0   -0.0141    0.0002   -0.1699
   -0.0141    0.0002   -0.1699    0.0048
    0.0260   -0.0024    0.0002   -0.0000

rank(ctrb) = 4  (need 4 for full controllability)

Output controllability matrix =
         0   -0.0141    0.0002   -0.1699
    0.0260   -0.0024    0.0002   -0.0000

rank(output ctrb) = 2  (need 2 for output controllability)
```

Published with MATLAB® R2025a

```
% Step 3
% PID controller design via pole placement

clear; clc;

% Parameters
R   = 0.35;
L   = 0.55;
ell = 0.85;
M   = 20;
m   = 90;
c1  = 1;
c2  = 10;
g   = 9.81;

J = M*L^2 + m*ell^2;

% State-space matrices (from Step 2)
A = [0, 1,                    0,        0;
     0, 0,                    1,        0;
     0, (M*g*L + m*g*ell)/J, -c1/J,     0;
     0, 0,                    0,        -c2/(M+m)];

B = [0; 0; -1/J; 1/(R*(M+m))];

F = [0; 0; m*g/J; 0];

% Controller design via pole placement
% The 4th pole is fixed at -c2/(M+m) since K(4)=0, so we place 3 poles
% on the 3x3 subsystem
A_sub = A(1:3, 1:3);
B_sub = B(1:3);

desired_poles = [-3+3i, -3-3i, -6];
K_sub = place(A_sub, B_sub, desired_poles);

K = [K_sub, 0];

disp('Gains [k1, k2, k3, 0]:'); disp(K);

% Verify closed-loop eigenvalues of the full 4x4 system
Acl = A - B*K;
disp('Closed-loop eigenvalues of A-BK:');
disp(eig(Acl));
```

```
Gains [k1, k2, k3, 0]:
   1.0e+03 *

   -7.6761   -4.6964   -0.8519         0

Closed-loop eigenvalues of A-BK:
  -0.0909 + 0.0000i
  -6.0000 + 0.0000i
```

```
-3.0000 + 3.0000i
-3.0000 - 3.0000i
```

Published with MATLAB® R2025a

```
% Step 4
% Trapezoidal disturbance profile and velocity simulation

clear; clc;

% Parameters
R   = 0.35;
L   = 0.55;
ell = 0.85;
M   = 20;
m   = 90;
c1  = 1;
c2  = 10;
g   = 9.81;

J = M*L^2 + m*ell^2;

% State-space matrices and controller (from Steps 2-3)
A = [0, 1,                    0,        0;
     0, 0,                    1,        0;
     0, (M*g*L + m*g*ell)/J, -c1/J,     0;
     0, 0,                    0,        -c2/(M+m)];

B = [0; 0; -1/J; 1/(R*(M+m))];

F = [0; 0; m*g/J; 0];

A_sub = A(1:3, 1:3);
B_sub = B(1:3);
K_sub = place(A_sub, B_sub, [-3+3i, -3-3i, -6]);
K = [K_sub, 0];

% Static-gain prediction for trapezoidal disturbance
d_peak = 0.01;
xdot_ss_predicted = m*g*d_peak / (R*c2);
fprintf('Static-gain prediction for d = %.3f m:\n', d_peak);
fprintf('  xdot_ss = %.4f m/s  (constraint: < 3 m/s)\n', xdot_ss_predicted);
fprintf('  Max d for xdot < 3 m/s: d_max = %.4f m\n', 3*R*c2/(m*g));
```

```
Static-gain prediction for d = 0.010 m:
  xdot_ss = 2.5226 m/s  (constraint: < 3 m/s)
  Max d for xdot < 3 m/s: d_max = 0.0119 m
```

Published with MATLAB® R2025a

```
% Step 5
% Full-state feedback via pole placement

clear; clc;

% Parameters
R   = 0.35;
L   = 0.55;
ell = 0.85;
M   = 20;
m   = 90;
c1  = 1;
c2  = 10;
g   = 9.81;

J = M*L^2 + m*ell^2;

% State-space matrices
A = [0, 1,                    0,         0;
     0, 0,                    1,         0;
     0, (M*g*L + m*g*ell)/J, -c1/J,     0;
     0, 0,                    0,         -c2/(M+m)];

B = [0; 0; -1/J; 1/(R*(M+m))];

F = [0; 0; m*g/J; 0];

% Full-state pole placement
% Dominant pair: ts = 4/sigma = 1 s  ->  sigma = 4
% Dominant: -4 +/- 4j  (zeta = 1/sqrt(2), omega_n = 4*sqrt(2))
% Non-dominant: -5, -6  (further into LHP but not too far, to keep gains
modest)
desired_poles = [-4+4i, -4-4i, -5, -6];
K = place(A, B, desired_poles);

disp('Gain vector K:'); disp(K);

% Verify closed-loop eigenvalues
Acl = A - B*K;
disp('Closed-loop eigenvalues of A-BK:');
disp(eig(Acl));
```

```
Gain vector K:
   1.0e+05 *

   -7.5055   -0.1591    0.5743    0.3184

Closed-loop eigenvalues of A-BK:
  -4.0000 + 4.0000i
  -4.0000 - 4.0000i
  -6.0000 + 0.0000i
```

```
-5.0000 + 0.0000i
```

Published with MATLAB® R2025a

```
% Step 6
% Torque saturation analysis

clear; clc;

% Parameters (same as Step 5)
R=0.35; L=0.55; ell=0.85; M=20; m=90; c1=1; c2=10; g=9.81;
J = M*L^2 + m*ell^2;

A = [0, 1, 0, 0;
     0, 0, 1, 0;
     0, (M*g*L + m*g*ell)/J, -c1/J, 0;
     0, 0, 0, -c2/(M+m)];
B = [0; 0; -1/J; 1/(R*(M+m))];
F = [0; 0; m*g/J; 0];

K = place(A, B, [-4+4i, -4-4i, -5, -6]);

% Saturation limit (80% of Step 5 peak)
Tmax_Step5 = 141;          % observed peak from Step 5
sat_limit  = 0.80 * Tmax_Step5;
fprintf('Saturation limit: +/- %.2f N*m (80%% of %g)\n', sat_limit,
Tmax_Step5);
```

```
Saturation limit: +/- 112.80 N*m (80% of 141)
```

Published with MATLAB® R2025a

```
% Step 7
% Steady-state analysis and static gain from d to xdot

clear; clc;

% Parameters (same as Steps 2-5)
R=0.35; L=0.55; ell=0.85; M=20; m=90; c1=1; c2=10; g=9.81;
J = M*L^2 + m*ell^2;

A = [0, 1, 0, 0;
     0, 0, 1, 0;
     0, (M*g*L + m*g*ell)/J, -c1/J, 0;
     0, 0, 0, -c2/(M+m)];
B = [0; 0; -1/J; 1/(R*(M+m))];
F = [0; 0; m*g/J; 0];

K = place(A, B, [-4+4i, -4-4i, -5, -6]);
Acl = A - B*K;

% Steady-state response to a unit step in d (d_ss = 1)
x_ss = -Acl \ F;

disp('Steady-state x for unit step in d:');
fprintf('  x1_ss (integral of theta) = %.4f\n', x_ss(1));
fprintf('  x2_ss (theta)             = %.4e rad\n', x_ss(2));
fprintf('  x3_ss (theta_dot)         = %.4e rad/s\n', x_ss(3));
fprintf('  x4_ss (xdot)              = %.4f m/s\n', x_ss(4));

fprintf('\nStatic gain d -> xdot: %.4f m/s per m of d\n', x_ss(4));

T_ss = -K * x_ss;
fprintf('\nSteady-state torque T_ss = %.4f N*m\n', T_ss);
fprintf('Predicted (m*g*d)        = %.4f N*m\n', m*g);
```

```
Steady-state x for unit step in d:
  x1_ss (integral of theta) = 10.7018
  x2_ss (theta)             = -0.0000e+00 rad
  x3_ss (theta_dot)         = -0.0000e+00 rad/s
  x4_ss (xdot)              = 252.2571 m/s

Static gain d -> xdot: 252.2571 m/s per m of d

Steady-state torque T_ss = 882.9000 N*m
Predicted (m*g*d)        = 882.9000 N*m
```

Published with MATLAB® R2025a

```
% Step 8
% Reduced 3-state model with full-state feedback

clear; clc;

% Parameters
R=0.35; L=0.55; ell=0.85; M=20; m=90; c1=1; c2=10; g=9.81;
J = M*L^2 + m*ell^2;

% 3-state model: x = [theta; theta_dot; xdot]
A = [0,                    1,      0;
     (M*g*L+m*g*ell)/J,  -c1/J,    0;
     0,                    0,     -c2/(M+m)];
B = [0; -1/J; 1/(R*(M+m))];
F = [0; m*g/J; 0];

% Controllability
Cctrb = ctrb(A, B);
fprintf('rank(ctrb) = %d  (need 3 for full controllability)\n\n',
rank(Cctrb));

% Pole placement for ts ≈ 1 s (dominant sigma = 4)
desired_poles = [-4+4i, -4-4i, -5];
K = place(A, B, desired_poles);

disp('Gain vector K:'); disp(K);

% Verify closed-loop eigenvalues
Acl = A - B*K;
disp('Closed-loop eigenvalues of A-BK:');
disp(eig(Acl));

% Steady-state (for discussion — comparison with Step 5)
d_ss = 0.01;  % same as Step 4 hold value
x_ss = -Acl \ (F * d_ss);
fprintf('\nFor a constant d = %g m:\n', d_ss);
fprintf('  theta_ss  = %.6f rad (%.4f deg)\n', x_ss(1), x_ss(1)*180/pi);
fprintf('  xdot_ss   = %.4f m/s\n', x_ss(3));
```

```
rank(ctrb) = 3  (need 3 for full controllability)

Gain vector K:
   1.0e+03 *

   -5.8229   -1.8208   -0.4898

Closed-loop eigenvalues of A-BK:
  -4.0000 + 4.0000i
  -4.0000 - 4.0000i
  -5.0000 + 0.0000i


For a constant d = 0.01 m:
```

```
theta_ss  = -0.009807 rad (-0.5619 deg)
xdot_ss   = 0.1174 m/s
```

Published with MATLAB® R2025a

```
% Step 10 (4-state case)
% Full-state observer design

clear; clc;

% Parameters
R=0.35; L_dim=0.55; ell=0.85; M=20; m=90; c1=1; c2=10; g=9.81;
J = M*L_dim^2 + m*ell^2;

% 4-state model (Step 5 setup)
A = [0, 1, 0, 0;
     0, 0, 1, 0;
     0, (M*g*L_dim+m*g*ell)/J, -c1/J, 0;
     0, 0, 0, -c2/(M+m)];
B = [0; 0; -1/J; 1/(R*(M+m))];
F = [0; 0; m*g/J; 0];

K = place(A, B, [-4+4i, -4-4i, -5, -6]);

% Observability check
C_theta = [0 1 0 0];
fprintf('C = [0 1 0 0] (theta only): rank(obsv) = %d\n', rank(obsv(A,
C_theta)));

C = eye(4);
fprintf('C = I: rank(obsv) = %d\n', rank(obsv(A, C)));

% Observer pole placement (5x faster than controller)
obs_poles = [-20+20i, -20-20i, -25, -30];
L = place(A', C', obs_poles)';

disp('Observer gain L:'); disp(L);
fprintf('Observer eigenvalues (A - L*C):\n'); disp(eig(A - L*C));
```

```
C = [0 1 0 0] (theta only): rank(obsv) = 2
C = I: rank(obsv) = 4
Observer gain L:
   20.0000  -19.0000         0         0
   20.0000   20.0000    1.0000         0
         0   12.0770   24.9859         0
         0         0         0   29.9091

Observer eigenvalues (A - L*C):
 -20.0000 +20.0000i
 -20.0000 -20.0000i
 -25.0000 + 0.0000i
 -30.0000 + 0.0000i
```

Published with MATLAB® R2025a

```
% Step 10 (3-state case)
% Full-state observer design for the 3-state model


clear; clc;

% Parameters
R=0.35; L_dim=0.55; ell=0.85; M=20; m=90; c1=1; c2=10; g=9.81;
J = M*L_dim^2 + m*ell^2;

% 3-state model (Step 8 setup)
A = [0,                    1,      0;
     (M*g*L_dim+m*g*ell)/J, -c1/J,  0;
     0,                    0,      -c2/(M+m)];
B = [0; -1/J; 1/(R*(M+m))];
F = [0; m*g/J; 0];

K = place(A, B, [-4+4i, -4-4i, -5]);

% Observability
C_theta = [1 0 0];
fprintf('C = [1 0 0] (theta only): rank(obsv) = %d\n', rank(obsv(A,
C_theta)));

C = eye(3);
fprintf('C = I: rank(obsv) = %d\n', rank(obsv(A, C)));

% Observer pole placement
obs_poles = [-20+20i, -20-20i, -25];
L = place(A', C', obs_poles)';

disp('Observer gain L:'); disp(L);
fprintf('Observer eigenvalues (A - L*C):\n'); disp(eig(A - L*C));
```

```
C = [1 0 0] (theta only): rank(obsv) = 2
C = I: rank(obsv) = 3
Observer gain L:
   20.0000  -19.0000         0
   32.0770   19.9859         0
         0         0   24.9091

Observer eigenvalues (A - L*C):
 -20.0000 +20.0000i
 -20.0000 -20.0000i
 -25.0000 + 0.0000i
```

Published with MATLAB® R2025a