AI Enabled Control Engineering

Lecture 10: State Observers for Control of the Rotary Inverted Pendulum

Professor Xun Huang

TA: Zhixiang Ju Haozhe Wang

Aeronautics and Astronautics
College of Engineering
Peking University

huangxun@pku.edu.cn
https://xunger99.github.io/xunger/

Recap: from Class 9 to Class 10

- In Class 9, we used LQR and MPC to stabilize the real rotary inverted pendulum near the upright equilibrium.
- The controller state was

$$x_k = [\theta_k, \dot{\theta}_k, \alpha_k, \dot{\alpha}_k]^\top.$$

- However, the hardware sensors directly measure only two angles:

$$y_k = [\theta_k, \alpha_k]^\top.$$

- The previous solution estimated angular velocities by low-pass-filtered finite differences.

Why do we need an observer?

- Direct angle difference is simple:

$$\dot{\theta}_k \approx \frac{\theta_k - \theta_{k-1}}{T_s}, \qquad \dot{\alpha}_k \approx \frac{\alpha_k - \alpha_{k-1}}{T_s}.$$

- But differentiation amplifies high-frequency sensor noise.
- Noisy velocity estimates may cause oscillatory PWM commands.
- An observer uses both the model and the measurements to estimate the full state.

Goal of Class 9

Replace low-pass finite-difference velocities by model-based observer states, then test LQR and MPC again on the real hardware.

Measured output and observer state

The local discrete model is

$$x_{k+1} = A_d x_k + B_d u_k.$$

The measured output is

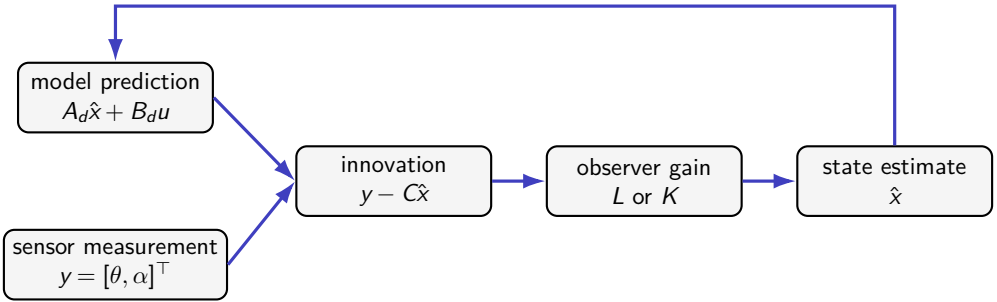
$$y_k = C x_k, \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The observer estimates

$$\hat{x}_k = \begin{bmatrix} \hat{\theta}_k & \widehat{\dot{\theta}}_k & \hat{\alpha}_k & \widehat{\dot{\alpha}}_k \end{bmatrix}^\top.$$

LQR and MPC will use $\hat{x}_k$, not the low-pass finite-difference state.

Observer as prediction plus correction



Intuition

The model predicts how the state should evolve. The measurement corrects the prediction whenever the predicted angle disagrees with the measured angle.

Observability check

For the pair (A_d, C), define the observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA_d \\ CA_d^2 \\ CA_d^3 \end{bmatrix}.$$

If

$$\operatorname{rank}(\mathcal{O}) = 4,$$

then the four-dimensional state can be reconstructed from the angle measurements and the model dynamics.

For the RIP system

Although $\dot{\theta}$ and $\dot{\alpha}$ are not directly measured, they can be estimated because the local model with outputs θ, α is observable.

Luenberger observer: discrete-time form

A simple discrete Luenberger observer can be written as

$$\hat{x}_k = (A_d - L_d C)\hat{x}_{k-1} + B_d u_{k-1} + L_d y_k.$$

Equivalently,

$$\hat{x}_k = A_d \hat{x}_{k-1} + B_d u_{k-1} + L_d \left(y_k - C\hat{x}_{k-1}\right).$$

- $A_d \hat{x}_{k-1} + B_d u_{k-1}$: model prediction.
- $y_k - C\hat{x}_{k-1}$: measurement innovation.
- L_d: observer gain to be designed.

Luenberger observer error dynamics

Define the estimation error

$$e_k = x_k - \hat{x}_k.$$

Using

$$x_k = A_d x_{k-1} + B_d u_{k-1},$$

and the observer update, we get

$$e_k = (A_d - L_d C) e_{k-1}.$$

Therefore, the observer is stable if

$$\rho(A_d - L_d C) < 1,$$

where $\rho(\cdot)$ is the spectral radius.

How to choose Luenberger observer poles

- Observer poles should be inside the unit circle.
- Faster poles give faster estimation convergence.
- But poles that are too fast amplify measurement noise.
- In practice, choose poles faster than the closed-loop control poles, but not extremely close to zero.

Suggested starting range

Start with real or complex poles with magnitudes around $0.5 \sim 0.85$, then test on LQR hardware logs.

Computing L_d by pole placement

For observer design, use the duality between controllability and observability:

$$\mathrm{eig}(A_d - L_d C) \quad \Longleftrightarrow \quad \mathrm{place}(A_d^\top, C^\top, \mathrm{poles}).$$

In Python:

```
from scipy.signal import place_poles

poles = [0.55+0.20j, 0.55-0.20j, 0.72, 0.82]
res = place_poles(Ad.T, C.T, poles)
Ld = res.gain_matrix.T
```

Student task

Try several pole sets, upload the resulting L_d, and compare observer-based LQR logs.

Luenberger observer tuning workflow

1. Choose a candidate pole set inside the unit circle.
2. Compute L_d using `place_poles`.
3. Fill L_d into `rip_lqr_luenberger_control.ino`.
4. Test LQR with observer state $\hat{x}_k$.
5. Compare $\alpha(t)$, PWM, and estimated velocities.
6. Keep the pole set that gives stable and smooth control.

Kalman observer: stochastic model

The discrete stochastic model is

$$x_{k+1} = A_d x_k + B_d u_k + w_k,$$

$$y_k = C x_k + v_k.$$

Here:

- w_k: process noise, representing model mismatch and unmodeled disturbances.
- v_k: measurement noise, representing sensor noise and quantization errors.

Assume

$$E[w_k w_k^\top] = Q_n, \qquad E[v_k v_k^\top] = R_n.$$

Kalman observer: prediction and correction

Prediction:

$$\hat{x}_{k|k-1} = A_d \hat{x}_{k-1|k-1} + B_d u_{k-1}.$$

Innovation:

$$\nu_k = y_k - C\hat{x}_{k|k-1}.$$

Correction:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \nu_k.$$

The Kalman gain is

$$K_k = P_{k|k-1} C^\top (C P_{k|k-1} C^\top + R_n)^{-1}.$$

Interpretation

Kalman filtering automatically balances model prediction and sensor measurement according to noise levels.

Steady-state Kalman observer

For a time-invariant system, the covariance can converge to a fixed matrix Π.

Then the Kalman gain becomes constant:

$$K = \Pi C^\top (C\Pi C^\top + R_n)^{-1}.$$

The embedded observer can be implemented as

$$\hat{x}_{k|k-1} = A_d \hat{x}_{k-1} + B_d u_{k-1},$$

$$\hat{x}_k = \hat{x}_{k|k-1} + K(y_k - C\hat{x}_{k|k-1}).$$

Why steady-state?

Only a fixed gain matrix is stored on STM32. No covariance matrix needs to be updated online.

How students tune Kalman parameters

- Choose process noise covariance Q_n.
- Choose measurement noise covariance R_n.
- Larger R_n: trust sensors less, smoother but slower correction.
- Larger Q_n: trust model less, faster response to measurement.
- Use logs to estimate sensor noise in the two measured angle channels.

Practical measurement noise estimate

Hold the system still, record θ and α, then compute the sample variances. These can be used as initial values for R_n.

Computing steady-state Kalman gain

In Python, students can use the discrete Riccati equation:

```
from scipy.linalg import solve_discrete_are

Pi = solve_discrete_are(Ad.T, C.T, Qn, Rn)
K = Pi @ C.T @ np.linalg.inv(C @ Pi @ C.T + Rn)
```

Then fill K into the STM32 firmware:

```
static const float K_KALMAN[4][2] = {
  { ... , ... },
  { ... , ... },
  { ... , ... },
  { ... , ... }
};
```

Student task

Try several Q_n, R_n settings and compare observer-based LQR logs first.

Experiment strategy of Class 9

1. Start from the LQR controller.
2. Replace low-pass velocity difference by a Luenberger observer.
3. Tune L_d until LQR hardware response is stable and smooth.
4. Replace the observer by a steady-state Kalman observer.
5. Tune Q_n, R_n until Kalman-based LQR performs well.
6. Then apply the good observer parameters to MPC.

Provided code packages

Package	Controller	State source
`class9_lqr_luenberger_lab.zip`	LQR	Luenberger observer
`class9_lqr_kalman_lab.zip`	LQR	steady-state Kalman observer
`class9_mpc_luenberger_lab.zip`	MPC	Luenberger observer
`class9_mpc_kalman_lab.zip`	MPC	steady-state Kalman observer

Important

Observer gain matrices are intentionally left as student parameters. Fill them before running the hardware experiment.

LQR observer experiment procedure

1. Calibrate `POT_ZERO`.
2. Compute LQR gain K, as in Class 8.
3. Design observer gain L_d or Kalman gain K.
4. Fill the gain matrix into the LQR observer firmware.
5. Upload firmware to STM32.
6. Hold the pendulum near upright and press Enter in Python.
7. Save CSV and compare with Class 8 low-pass-difference LQR logs.

MPC observer experiment procedure

1. Use the observer parameters that worked well in LQR.
2. Fill the same observer matrix into the MPC observer firmware.
3. Upload MPC observer firmware to STM32.
4. Hold the pendulum near upright and press Enter in Python.
5. Compare MPC-Luenberger and MPC-Kalman logs.

What to compare in the report

- Low-pass-difference LQR from Class 8.
- Low-pass-difference MPC from Class 8.
- Luenberger-observer LQR.
- Kalman-observer LQR.
- Luenberger-observer MPC.
- Kalman-observer MPC.

Recommended plots:

$$\alpha(t), \quad \theta(t), \quad \widehat{\dot{\theta}}(t), \quad \widehat{\dot{\alpha}}(t), \quad u(t).$$

Evaluation

A good observer should give smooth velocity estimates, stable upright control, and no long-term PWM saturation.

Summary of Class 10

- The RIP sensors directly measure only θ and α.
- Direct finite difference is simple but amplifies noise.
- Luenberger observers use pole placement to make estimation error converge.
- Steady-state Kalman observers use noise covariance matrices to balance model prediction and sensor correction.
- In the experiment, we first tune observers with LQR, then reuse good observers in MPC.

Take-home message

Observers are the bridge between limited hardware measurements and full-state optimal control.