

Lecture 13

- State Estimation
- Open-loop state estimation
- Closed-loop state estimation
 - Error dynamics
 - Guidelines for setting estimator poles
- Separation Principle
 - Using the estimation in feedback: $u = -K\hat{x}$

Two reasons for state estimation:

1. Not all states are measured directly (e.g., encoder tracks cart position; velocity not measured)
2. Measurements may have noise (e.g., sensor noise, or round-off error due to quantization, etc.)

We want the state for our feedback, $u = -Kx$.

Can we use “estimate” of state, via: $u = -K\hat{x}$?

$$Y = CX + DU \quad (\text{if no noise in measurements})$$

↙ AKA: Observer design.

State Estimation - Problem Statement

Sometimes, feedback control requires we “estimate” some of the states because, for example:

1. We might not sense all states in the output vector y , e.g.:

$$X = \begin{bmatrix} \dot{x} \\ x \end{bmatrix} \quad y = [0 \quad 1]X + [0]u$$

~~position~~ $y = \text{position}$ (velocity not measured)

2. Our measurements might be noisy, e.g.:

$$Y = CX + DU + V$$

measurement noise

PROBLEM STATEMENT: If we start with some “initial value” as an ESTIMATE of the state vector (for $k=0$), how can ESTIMATE the states for $k>0$ (over time)?

Before: “state feedback”: $u = -Kx$

Now: x is not perfectly known,

$$\boxed{u = -K\hat{x}}$$

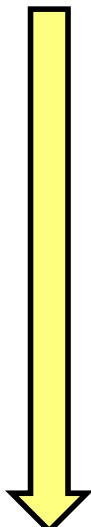
\hat{x} : est. of x
use an estimate

Approach 1: Mimic the known dynamics

- Make an initial guess, \hat{x} , of the ($n \times 1$) initial state, x .
 - *There will likely be some error in this guess...*
- Run a “digital twin”, simulating **linearized dynamics**.
- Assume u is known perfectly: you are setting it.

Below, there is no noise.

- We will revisit effects of noise later...



State Estimation - “Open loop” approach

Start with some “initial value” as an ESTIMATE of the states...
...and assume this vector of value evolves via “ $Ax+Bu$ ” dynamics.

e_x :
error



$$\text{State: } x(k+1) = A_d x(k) + B_d u(k)$$

$$\text{Estimate of state: } \hat{x}(k+1) = A_d \hat{x}(k) + B_d u(k)$$

common alternate
notation for “error”

$$\text{Error in the estimate: } e_x(k+1) = x(k+1) - \hat{x}(k+1) \quad \text{or: } \tilde{x}(k+1)$$

notation for error in our lectures

$$e_x = x - \hat{x} \text{, so...}$$

$$x_{K+1} = A_d x_K + B_d u_K$$

$$-(\hat{x}_{K+1} = A_d \hat{x}_K + B_d u_K)$$

$$(x_{K+1} - \hat{x}_{K+1}) = A_d(x_K - \hat{x}_K) + B_d(u_K - u_K) \xrightarrow{0}$$

$$e_{K+1} = A_d e_K \quad \text{← same poles as open-loop system.}$$

What happens?

The vector of errors, $e_x = x - \hat{x}$, has dynamics identical to the actual state.

i.e., whatever the initial conditions are for the error vector at $k=0$, they evolve over time as: $ze_x = A_d e_x$

Approach 1: What happens?

Instead of n states, we now have $2n$ states:

- The original ($nx1$) vector x
- Plus, an ($nx1$) vector, $e_x(k)$, of errors

Example (below, at right):

- State x has some initial condition
- Error, $e_x = x - \hat{x}$, has its own initial condition

$$(x_{k+1} - \hat{x}_{k+1}) = A_d(x_k - \hat{x}_k) + B_d(u_k - u_e)^0$$

$$e_{k+1} = A_d e_k \quad \text{← same poles as open-loop system.}$$

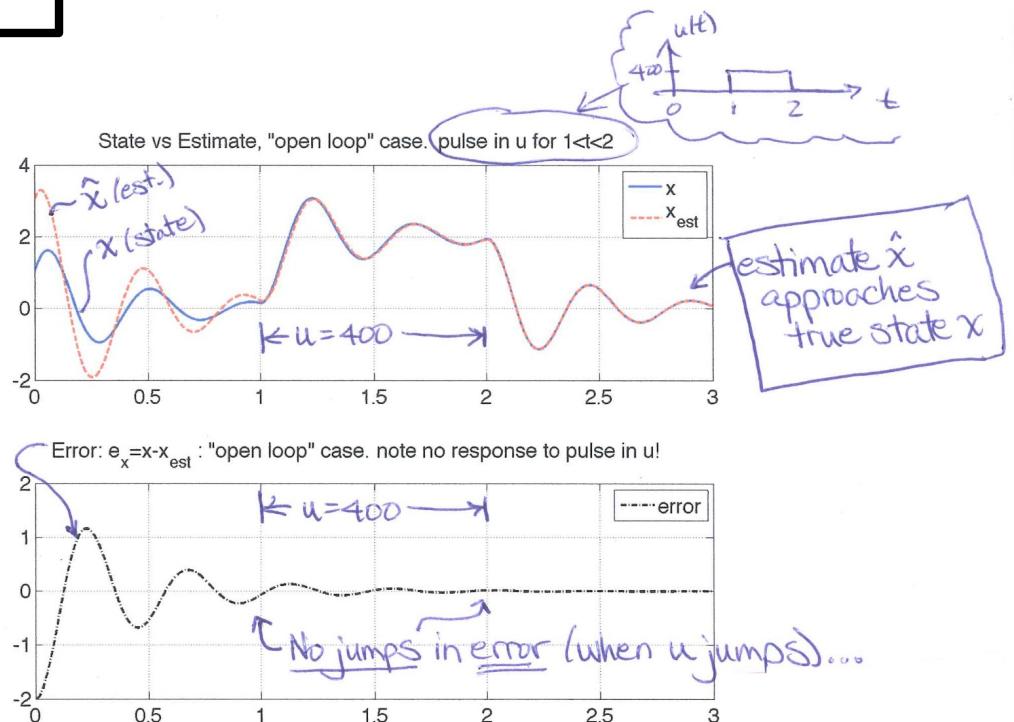
In top subplot, the position variable of a 2nd-order system decays via stable, underdamped poles of matrix A.

- The state, x , is perturbed by the “pulse” in input, u

In bottom subplot, the error has the same dynamics, given by $\text{eig}(A)$.

- The error IS NOT PERTURBED by the input u , because we know what u is.

Recall from last page: $ze_x = A_d e_x$



Approach 2: Use a feedback signal

For feedback, we only have access to output y via: $y = Cx + Du$.

- So, we might not have full access to states in x .
- But: can't we reason about what output we expect to measure, if our estimate is correct... ??

What to do :

- Compare what we have access to, y , to what we expect that output to be, based on the estimate: $\hat{y} = C_d \hat{x} + D_d u$
 - Scale this, to add in a feedback “correction factor” that is: $L_p(y_k - \hat{y}_k)$
-

State Estimation - “feedback” approach

Start with some “initial value” as an ESTIMATE of the states...
...use BOTH a MODEL (“ $Ax+Bu$ ”) and a MEASUREMENT (“ $y=Cx+Du$ ”).

Estimate of state WITH MEASUREMENT FEEDBACK : $\hat{x}(k+1)$

But how do we “feed back” MEASUREMENT info?

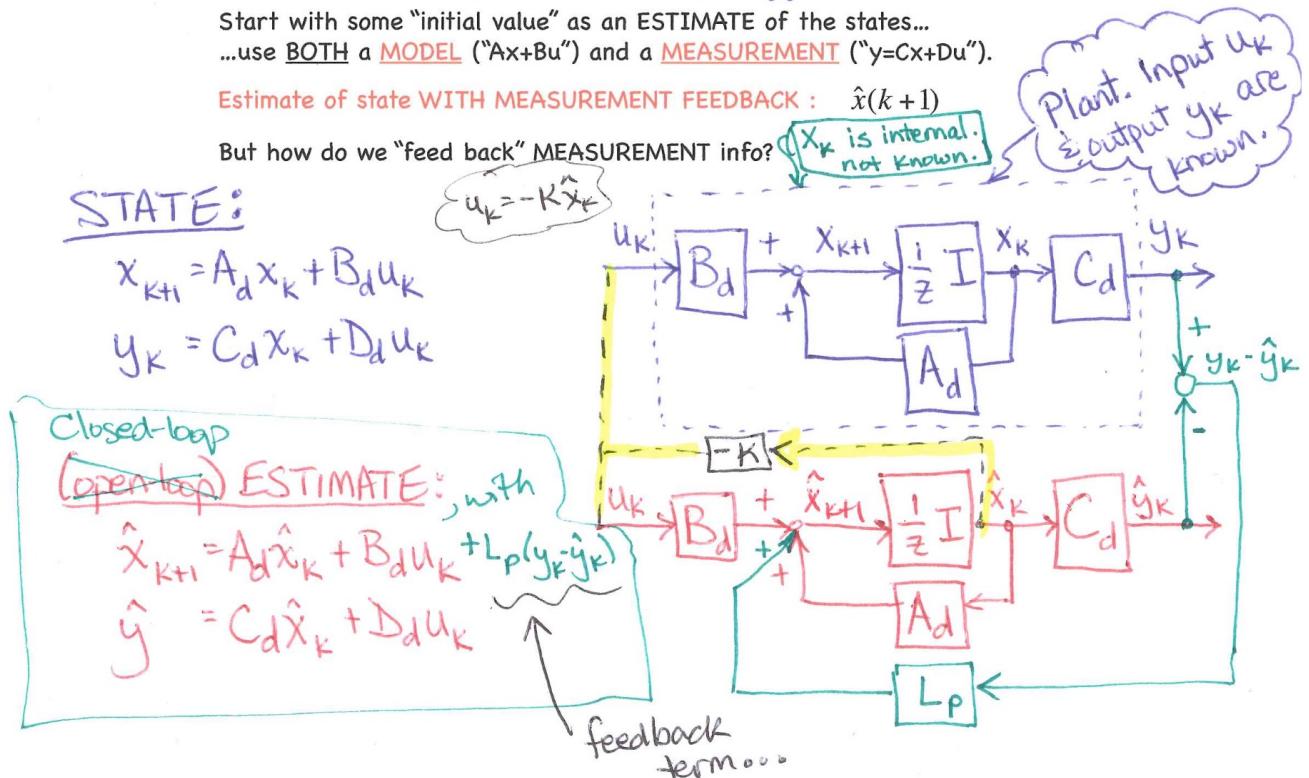
At top: The linear system for the actual state dynamics.

STATE:

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

$$y_k = C_d x_k + D_d u_k$$

Below: The “digital twin” tracks an estimate of the state.



L_p times the difference between measured and expected outputs provides a feedback signal, to drive the error in estimate to zero.

Approach 2: Use a feedback signal

State Estimation - "feedback" approach

Start with some "initial value" as an ESTIMATE of the states...

...use BOTH a MODEL (" $Ax+Bu$ ") and a MEASUREMENT (" $y=Cx+Du$ ").

Estimate of state WITH MEASUREMENT FEEDBACK : $\hat{x}(k+1)$

But how do we "feed back" MEASUREMENT info?

x_k is internal.
not known.

Plant. Input u_k
& output y_k are
known.

STATE:

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

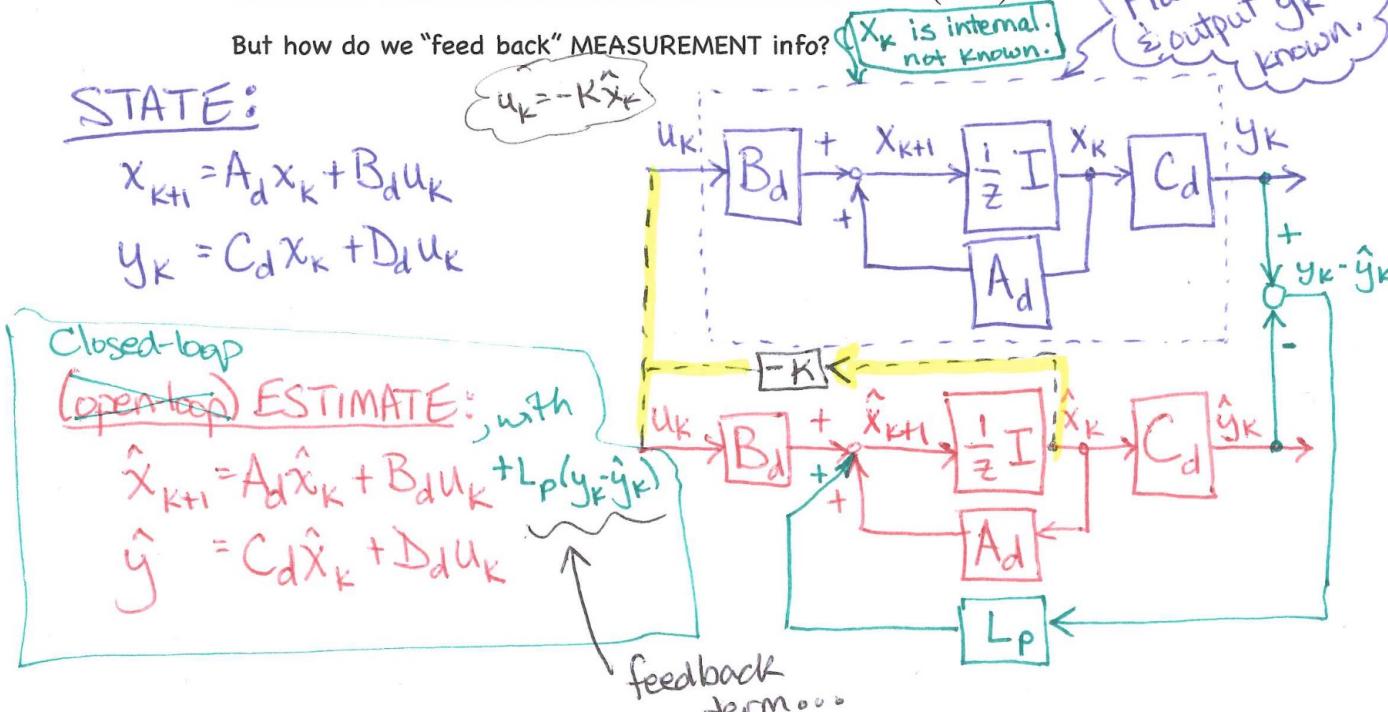
$$y_k = C_d x_k + D_d u_k$$

Closed-loop

(open-loop) ESTIMATE: with

$$\hat{x}_{k+1} = A_d \hat{x}_k + B_d u_k + L_p(y_k - \hat{y}_k)$$

$$\hat{y} = C_d \hat{x}_k + D_d u_k$$



L_p is a gain matrix, for feedback in the estimation problem.

Approach 2: What happens, mathematically?

Now with feedback:

The additional **error states** have their own dynamics over time, given by: $\text{eig}(A - L_p C)$.

So, the initial error decays (recall, assuming no noise so far) via those poles: $\text{eig}(A - L_p C)$, ...so long as those poles are stable.

This is it, for the error dynamics.

However, the error acts as a signal input to affect the actual states.

(To be discussed more later...)

State Estimation - "feedback" approach

(Continued...)

$$\begin{aligned} \text{State: } zX &= x_{k+1} = A_d x_k + B_d u_k \\ \text{estimate: } z\hat{x} &= \hat{x}_{k+1} = \underbrace{A_d \hat{x}_k + B_d u_k}_{\text{Error dynamics?}} + L_p C(x_k - \hat{x}_k) \\ e(k+1) &= x_{k+1} - \hat{x}_{k+1} = A_d(x_k - \hat{x}_k) + D - L_p C(x_k - \hat{x}_k) \\ &= (A_d - L_p C)(x_k - \hat{x}_k) \end{aligned}$$

$$e_{k+1} = (A - L_p C)e_k$$

(Analogous to
 $\text{eig}(A - Bk)$
C.L. dyn.)



closed-loop dynamics of
the error, $e_k = x_k - \hat{x}_k$,
are set by poles: $\text{eig}(A - LC)$

Estimator poles

$$\begin{cases} y_k = Cx_k \\ \hat{y}_k = C\hat{x}_k \end{cases}$$

Approach 2: Predictor vs Current Estimator

On a computer, it takes finite time to compute the estimate.

So, we're usually solving for what is technically called the “predictor” estimator gains, in L_p .

(i.e., are we predicting x_k or are we predicting x_{k+1} ?)

L_p is the “predictor” estimator gain matrix.
It is used in feedback to predict $x(k+1)$.

L_c is the “current” estimator gain matrix.
It is used to predict $x(k)$.

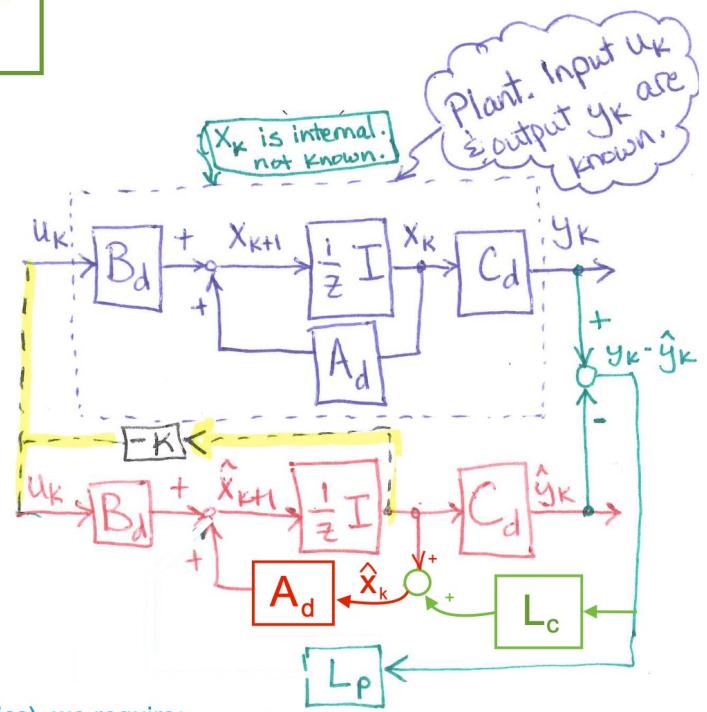
Let us “cut” the signal from L_p .

At right is where the feedback from L_c enters the block diagram.

Note that $\hat{x}(k)$ has now moved, since it is defined only AFTER the feedback contribution (from L_c) is added in.

For the same closed loop poles (i.e., dynamics), we require:

$$L_p = A_d * L_c, \quad \text{so} \quad L_c = \text{inv}(A_d) * L_p$$



Approach 2: Pole placement (applied with new details...)

Looks similar to pole placement?

Let's compare:

- pole placement for state feedback

Recall: $x_{k+1} = A_d x_k + B_d u_k \leftarrow \omega$ $u_k = -K x_k$

$$x_{k+1} = (A_d - B_d K) x_k \leftarrow \text{pick } K \text{ to set } \text{eig}(A - BK) \text{ (i.e., C.L. poles...)}$$

- pole placement for estimator design

$$e_{k+1} = (A_d - L_p C_d) e_k \leftarrow \text{Now, pick } L_p \text{ to set } \text{eig}(A - LC) = \text{eig}(A - LC)^T$$

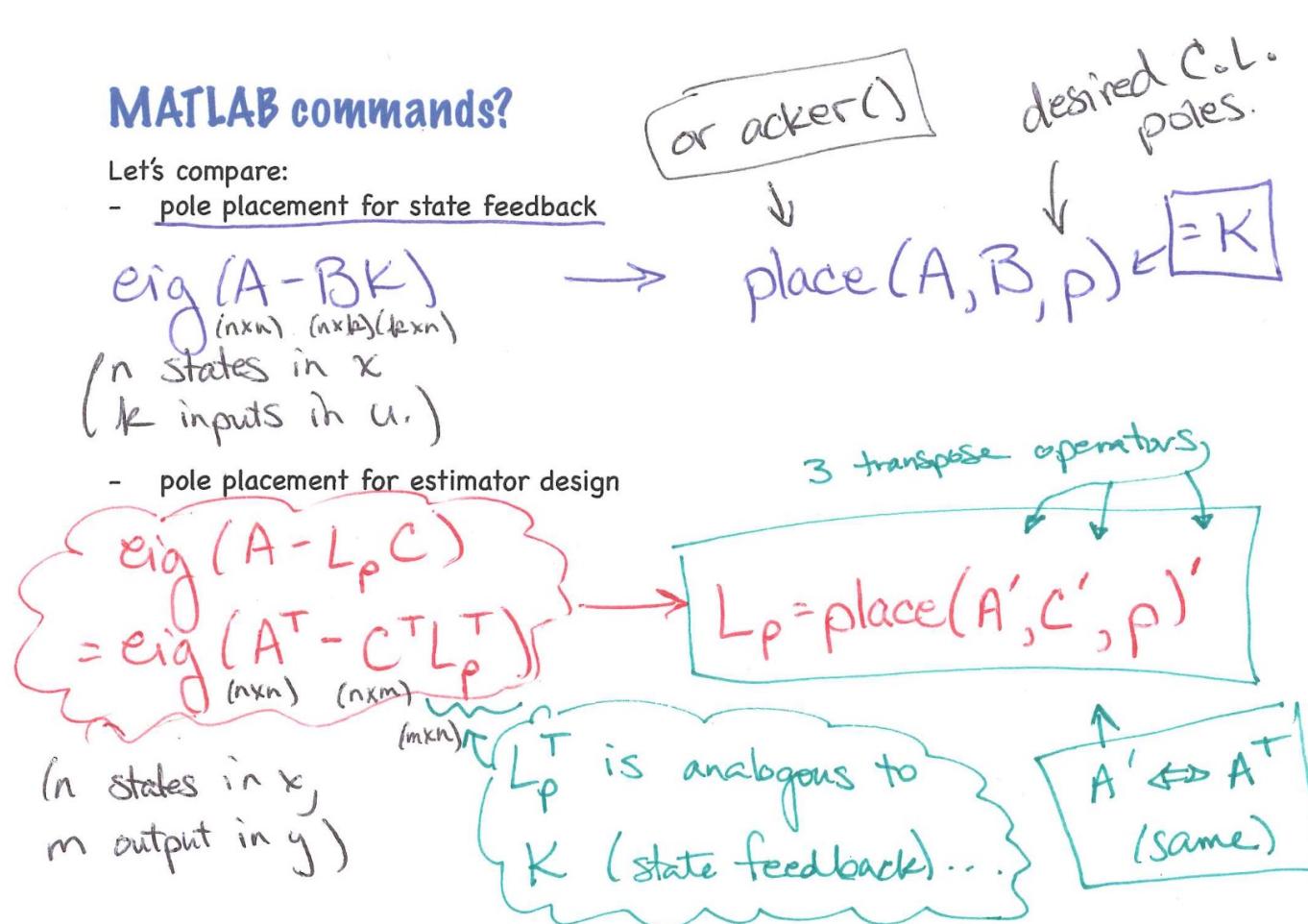
L_p : observer feedback, [note: $(A - LC)^T = A^T - C^T L^T$] driving estimate errors to zero, w/ desired "dynamics".

Approach 2: Matlab commands

For control, $K = \text{place}(A, B, pc_des)$

For estimation, $L_p = \text{place}(A', C', pe_des)'$

Note there are **three (3) transpose characters**, for the command above.



Approach 2: Observability is similar to Controllability

Basically, we replace:
with:

$$\begin{array}{ll} A \text{ and } B \\ A^T \text{ and } C^T \end{array}$$

Then, make sure the corresponding matrix is “full rank” (to set n poles, via n free variables).

Observability (vs Controllability)

→ When can we set estimator poles arbitrarily?
“Controllability” question is ANALOGOUS to “Observability”

$$C = [B, AB, A^2B, \dots, A^{n-1}B]$$

$$C_o = \text{ctrb}(A, B)$$

$$K = \text{acker}(A, B, P)$$

eig(A-BK) → poles of closed-loop plant.

$$\tilde{O}^T = [C^T, A^T C^T, (A^T)^2 C^T, \dots, (A^T)^{n-1} C^T]$$

Observability matrix

$$\tilde{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

$$Ob = \text{obsv}(A, C)$$

typically used
in 147B.

$$\begin{aligned} L_p &= \text{acker}(A', C', P') \\ L_c &= \text{acker}(A' A^* C', P') \end{aligned}$$

Here is how to calculate the predict gain matrix (L_p) versus the current predictor (L_c) in MATLAB:

Approach 2: How to pick estimator poles..

...but how do you choose estimator poles?

1. Want “error dynamics” to be “faster” than controlled state dynamics.
 - i.e., fast decay; minimal overshoot

2. Do not want large gains to amplify sensor noise.
 - $Y = CX + n$, where “n” is sensor noise

Approach 2: Guidelines, to set estimator poles

Gain matrix L_p sets poles (i.e., dynamics) of the estimator error.

Here is the “feedback correction” to the estimate:

$$z\hat{x} = A\hat{x} + Bu + L_p(y - \hat{y})$$

Now, assume:

$$y = Cx + Du + n$$


Add noise to y .

where n is sensor noise.

Larger gains in L_p will generally result in:

- FASTER poles, so error dies away more rapidly,
- But also AMPLIFICATION of the sensor noise!

→ Trade-off!

Typically, aim for estimator poles
2x-4x “faster” than $\text{eig}(A-BK)$,
without too much overshoot.

This issue is addressed more formally by “Kalman filtering”, in which L_p is set in an optimal way to minimize noise in the estimate.
(This topic is scheduled to be presented in Lecture 16...)

Separation Principle

Question:

If we do not use the true state but instead use the estimate for feedback via $u = -K\hat{x}$, what happens? (*What are the poles? Is it stable?*)

Brief Answer:

The error dynamics are given only by $\text{eig}(A - L_p C)$.

The state dynamics end up being “driven” by the error (as a signal), in addition to having their own closed-loop poles given by $\text{eig}(A - BK)$.

So, the entire system now has twice as many poles (i.e., $2n$ poles), which are the combination of both:
 $\text{eig}(A - L_p C)$
 $\text{eig}(A - BK)$

“full rank”
When rank is n

Observability and Controllability

→ When can we set estimator poles arbitrarily?

“Controllability” question is ANALOGOUS to “Observability”

$$C_i = [B, AB, \dots, A^{n-1}B]_{n \times [n \times m]}^{\text{rank } n} \quad \text{c } m \text{ inputs}$$

$$\textcircled{1} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}_{([k \times n] \times n)}$$

Plant dynamics

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

$$\text{state feedback: } \tilde{u}_k = -K x_k$$

$$\therefore x_{k+1} = (A - BK)x_k$$

$\text{eig}(A - BK)$ set C.L. poles
Controllability (linear case); Can set n C.L. poles arbitrarily

State estimate

$$\hat{x}_{k+1} = A_d \hat{x}_k + B_d \tilde{u}_k + L_p (y_k - \hat{y}_k)$$

$$\text{Error: } e_{k+1} = x_k - \hat{x}_k$$

$$e_{k+1} = (A - L_p C)e_k$$

error dynamics (for est. error)
 $\text{eig}(A - L_p C)$ are poles for error dyn.
Observability; Can set n poles for error dyn.

Separation Principle

Observability

Example

$$\text{DT} \quad A = \begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$X = \begin{bmatrix} \dot{x} \\ x \end{bmatrix} \quad \begin{array}{l} \text{velocity} \\ \text{position} \end{array}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad \begin{array}{l} \text{single} \\ \text{input} \end{array}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad \begin{array}{l} \text{single} \\ \text{output} \end{array}$$

DT → assume ZOH + Sampling
 $\Rightarrow T = 0.1 \text{ sec}$

$$A_d \approx \begin{bmatrix} -0.90016 & -0.09500 \\ -0.09500 & -0.99517 \end{bmatrix}$$

$$B_d \approx \begin{bmatrix} 0.09500 \\ 0.0048 \end{bmatrix}$$

$$C_d = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Given n measurements, for an n^{th} -order system $(y_1 \rightarrow y_n)$,
 Can we solve mathematically for all n states in $X_1, (x_1, x_2, \dots, x_n)$?

Assume x_1 is not known, but $y_k \in u_k$ are, for $\forall k$...

$$\textcircled{1} \quad y_1 = CX_1 = [0 \ 1] \begin{bmatrix} \dot{x}_1 \\ x_1 \end{bmatrix} = x_1$$

$$\textcircled{2} \quad y_2 = CX_2 = C(A_d X_1 + B_d u_1)$$

matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} C \\ CA_d \end{bmatrix} X_1 + \begin{bmatrix} 0 \\ CB_d u_1 \end{bmatrix}$$

Can solve iff $\begin{bmatrix} C \\ CA_d \end{bmatrix}$ is invertible,
 i.e., Full Rank

$$X_1 = \left(\begin{bmatrix} C \\ CA_d \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} 0 \\ CB_d u_1 \end{bmatrix} \right)$$

Separation Principle

Recall STATE FEEDBACK

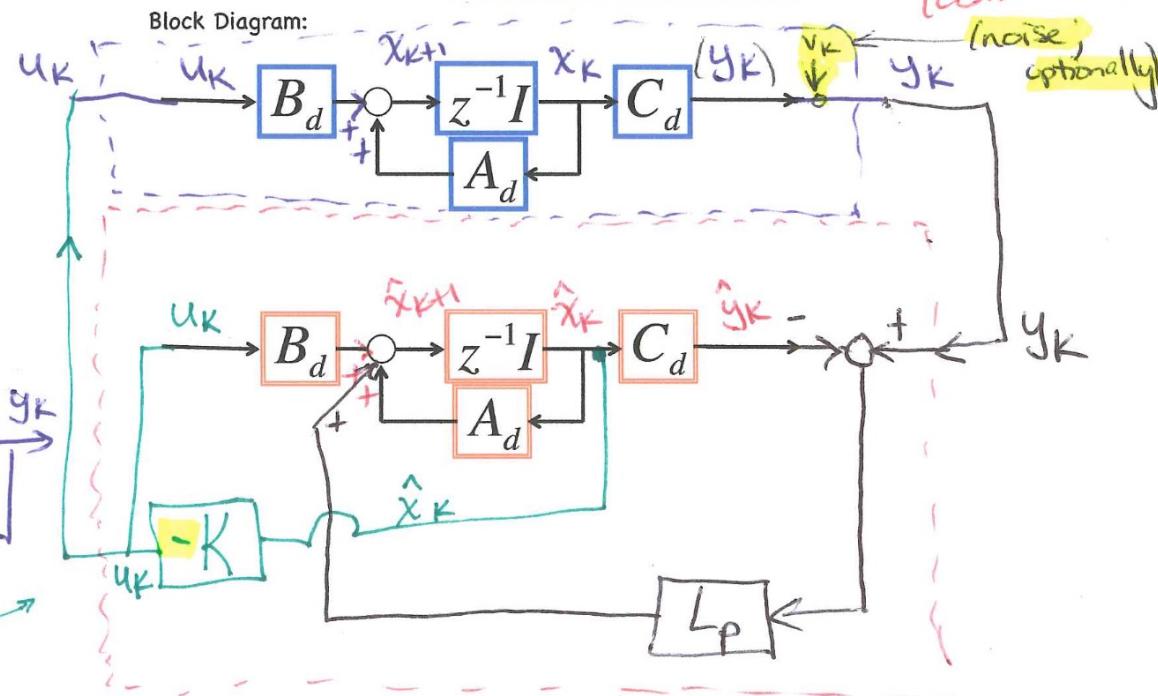
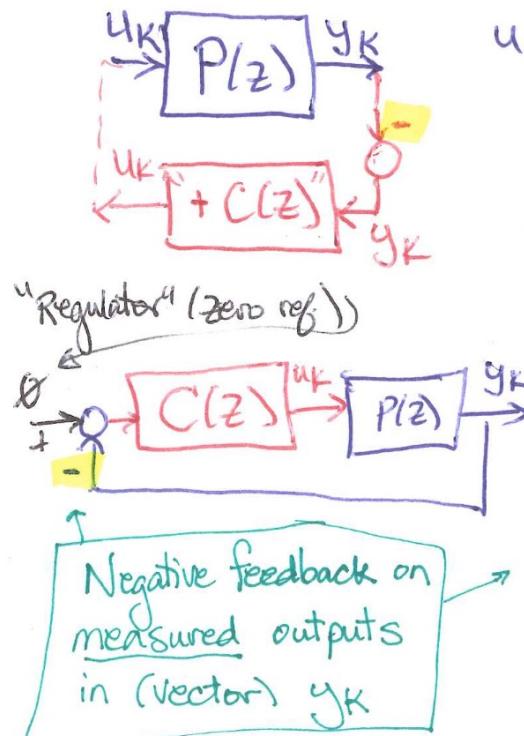
Control uses the true state:

$$u(k) = -Kx(k)$$

What if we use the state ESTIMATE for feedback control, instead?

$$u(k) = -K\hat{x}(k)$$

uses \hat{x} (estimate)



Separation Principle: Closed-loop dynamics

$$e = x - \hat{x}$$

Closed-loop Dynamics, for state (x) and error (e_x)

Note: $u_k = -K\hat{x}_k$ depends on the estimate, \hat{x}_k .

So, how does this (using \hat{x} and not x) affect plant dynamics? $u = -K\hat{x} = -Kx + Ke$

$$\textcircled{1} \quad x_{k+1} = A_d x_k + B_d (-Kx_k + Ke_k)$$

$$\textcircled{1} \quad x_{k+1} = (A_d - B_d K) x_k + B_d K e_k$$

MATRIX form for DYNAMICS:

$$\begin{bmatrix} x_{k+1} \\ e_{k+1} \end{bmatrix} = \begin{bmatrix} (A_d - B_d K) & BK \\ \Phi & (A_d - L_p C) \end{bmatrix} \begin{bmatrix} x_k \\ e_k \end{bmatrix}$$

Total system now has
2n "states": $x \in \mathbb{R}^{(nx1)}$, $e \in \mathbb{R}^{(nx1)}$

$$\textcircled{2} \quad e_{k+1} = x_{k+1} - \hat{x}_{k+1}$$

$$= (A x_k + B u_k) - (A \hat{x}_k + B u_k + L_p C(x_k - \hat{x}_k))$$

$$e_{k+1} = A x_k - A \hat{x}_k + L_p C(x_k - \hat{x}_k)$$

$$\textcircled{2} \quad e_{k+1} = (A - L_p C)e_k$$

eigenvalues?

$$\text{eig}\{(A - BK), BK\}$$

which are $\text{eig}(A - BK) \cup \text{eig}(A - LC)$

Separation Principle : Closed-loop dynamics

MATRIX form for DYNAMICS:
eigenvalues?

$$\begin{bmatrix} x_{k+1} \\ e_{k+1} \end{bmatrix} = \begin{bmatrix} (A - BK) & BK \\ 0 & (A - LC) \end{bmatrix} \begin{bmatrix} x_k \\ e_k \end{bmatrix}$$

NOTE - Matrix above has "Upper Block Diagonal" form.
Examples:

$n=1$ →
state
 $(2n=2)$

$$\det \left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} a & c \\ 0 & b \end{bmatrix} \right) = 0$$

Dynamics of states & errors
($e \equiv x - \hat{x}$) together all have poles given by:

$$\text{eig}(A - BK) \cup \text{eig}(A - LC)$$

(union)

$$\det \left(\begin{bmatrix} (z-a), -c \\ 0, (z-b) \end{bmatrix} \right) = 0 = (z-a)(z-b) - 0$$

$$\boxed{z=a} \text{ and } \boxed{z=b}$$

are the poles.

$n=2$ States
 $(2n=4)$

$$\det \left(\begin{bmatrix} z & 0 & 0 & 0 \\ 0 & z & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & z \end{bmatrix} - \begin{bmatrix} a_1 & a_2 & c_1 & c_2 \\ a_3 & a_4 & c_3 & c_4 \\ 0 & 0 & b_1 & b_2 \\ 0 & 0 & b_3 & b_4 \end{bmatrix} \right) = 0$$

$(4 \times 4) \quad (4 \times 4)$

See next page ...

Separation Principle : Examples

Eigenvalue examples, via MATLAB

Below is code to "see" some of the ideas for yourself - at least for our toy examples:

```

syms a1 a2 a3 a4 b1 b2 b3 b4 c1 c2 c3 c4 z
A = [a1 a2; a3 a4];
B = [b1 b2; b3 b4];
C = [c1 c2; c3 c4];
M = [A C; zeros(2,2) B]
M =
[ a1, a2, c1, c2]
[ a3, a4, c3, c4]
[ 0, 0, b1, b2]
[ 0, 0, b3, b4]
zI = z*eye(4)
zI =
[ z, 0, 0, 0]
[ 0, z, 0, 0]
[ 0, 0, z, 0]
[ 0, 0, 0, z]
det(zI-M)
ans =
z^4 - a4*z^3 - b1*z^3 - b4*z^3 - a1*z^3 + a1*a4*z^2 - a2*a3*z^2 + a1*b1*z^2 + a1*b4*z^2
+ a4*b1*z^2 + a4*b4*z^2 + b1*b4*z^2 - b2*b3*z^2 + a1*a4*b1*b4 - a1*a4*b2*b3 -
a2*a3*b1*b4 + a2*a3*b2*b3 - a1*a4*b1*z + a2*a3*b1*z - a1*a4*b4*z + a2*a3*b4*z -
a1*b1*b4*z + a1*b2*b3*z - a4*b1*b4*z + a4*b2*b3*z
simplify(det(zI-M))
ans =
(a2*a3 - a1*a4 + a1*z + a4*z - z^2)*(b2*b3 - b1*b4 + b1*z + b4*z - z^2)

```

$$\det \left(\begin{bmatrix} z & 0 & 0 & 0 \\ 0 & z & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & z \end{bmatrix} - \begin{bmatrix} a_1 & a_2 & c_1 & c_2 \\ a_3 & a_4 & c_3 & c_4 \\ 0 & 0 & b_1 & b_2 \\ 0 & 0 & b_3 & b_4 \end{bmatrix} \right) = \det(zI - M) = 0$$

↑ A ↓ B ← "Upper block diagonal"

$$\text{eig}(A) \rightarrow \det(zI - A) = (z - a_1)(z - a_4) - (-a_3)(-a_2) = 0$$

$$z^2 - (a_1 + a_4)z + (a_1a_4 - a_2a_3) = 0$$

$$\text{eig}(B) \rightarrow z^2 - (b_1 + b_4)z + (b_1b_4 - b_2b_3) = 0$$

"Brute force" solution via Matlab...
...matches union of eig(A) \ eig(B)

Separation Principle : The Big Take-Away...

Separation Principle

For feedback with an ESTIMATOR,

$$U_K = -K \hat{x}_K$$

, when $\hat{x}_{K+1} = A \hat{x}_K + B u_K + L_p (y - C \hat{x}_K)$

$$y = C x$$

1) CONTROL for nth-order system has n poles, determined by $\text{eig}(A-BK)$.

First: Pick (dominant) closed-loop poles for $\text{eig}(A-BK)$,
(eig., via pole placement).

2) ESTIMATION of n states results in another n poles, determined by $\text{eig}(A-LC)$

Then: Pick estimator poles to be 2x-4x "faster"
than $\text{eig}(A-BK)$, so they do not dominate dynamics.

3) TOTAL system now has $2n$ poles:

? they are the union of $\text{eig}(A-BK)$ & $\text{eig}(A-LC)$.

i.e., we can separate design of all $2n$ poles

via two uncoupled (sets of) equations.

Separation
Principle

Separation Principle : Practical Strategy (to place poles)

How to place poles, to set "K" and L_p ?

Controller poles:

Use classic control design specifications, (from 147a)
 CI (s-plane) DT (z-plane)

ω_{d_1} , t_{rise} , M_p , etc



DT (z-plane)

$$Z = e^{ST}$$

Estimator poles:

Why $2x - 4x$ faster than $eig(A-BK)$? Ans: $eig(A-BK)$ should dominate (meaning have "slowest" decay rates), AND...

Effects of Noise?

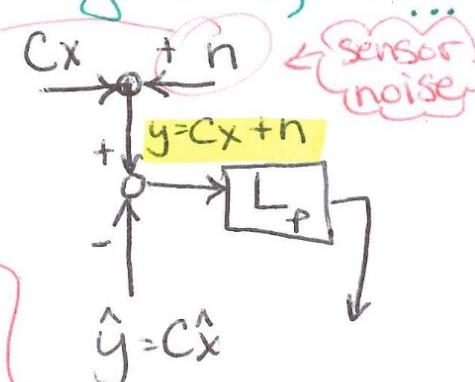
$$\begin{bmatrix} x_{k+1} \\ e_{k+1} \end{bmatrix} = \begin{bmatrix} (A - BK) & BK \\ 0 & (A - LC) \end{bmatrix} \begin{bmatrix} x_k \\ e_k \end{bmatrix} + \begin{bmatrix} 0 \\ -L \end{bmatrix} n_k$$

$$x_{k+1} = Ax_k + Bu_k$$

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L_p(Cx_k + n_k - C\hat{x}_k)$$

$$e_{k+1} = x_k - \hat{x}_k = A(x_k - \hat{x}_k) - L_p(Cx_k - C\hat{x}_k + n_k)$$

$$\therefore e_{k+1} = (A - L_p C)e_k - L_p n_k$$



Derivation...

Separation Principle : First, set DT controller poles...

Example: control and estimation, w/ MATLAB

Control Canonical Form...

$$A = \begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$p_{ct} = [-10 + 10j, -10 - 10j]$$

$$A - BK = \begin{bmatrix} -1 - K_1 & -1 - K_2 \\ 1 & 0 \end{bmatrix}$$

or design

CT (continuous time) - n states (x), m inputs (u), l outputs (y)

DT (discrete time), $[T = 0.01 \text{ sec}]$ - system resulting from ZOH

$$K = [19, 199].$$

$$M = \expm([A, B; \text{zeros}(m, n+m)] * T)$$

$$Ad = M(1:n, 1:n);$$

$$Bd = M(1:n, n+1:end);$$

$$\left\{ A_d \approx \begin{bmatrix} .99 & -.01 \\ .01 & 1.0 \end{bmatrix}, B_d \approx \begin{bmatrix} .01 \\ 0 \end{bmatrix} \right.$$

$$pdt = \exp(pct*T);$$

$$Z = e^{sT}$$

$$Kd = \text{place}(Ad, Bd, pdt)$$

$$\leftarrow \text{Gives } K_d = [18.12, 180.88]$$

DT design

DT gains
 K_d : Similar but not exactly the same as CT gains in K_{ct}

Separation Principle : Set estimator poles 2x to 4x faster

Below, pct are the closed-loop poles of $\text{eig}(A-BK)$.

`pe_ct` are the CT closed-loop poles of $\text{eig}(A-LC)$. (i.e., estimator poles)

`pe_dt` are the DT estimator (closed-loop) poles.

```

Example: continued

C = [0 1];
pe_ct = 3*pct; ← Es
pe_dt = exp(pe_ct*T);
Lpc = place(A', C', pe_ct)';
Lpd = place(Ad', Cd', pe_dt)';
Kd = place(Ad, Bd, pdt)

```

Estimator poles, 2x-4x faster.

X_0
 \hat{X}_0
 X_{k+1}
 X_{k+1}

```

Kd = place(Ad, Bd, pdt)
Xinit = [10;10];
Xest = [0;0];
N = 100
X = zeros(2,N); X(:,1) = Xinit;
Xe = X; Xe(:,1) = Xest;
u = zeros(1,N);
for n=1:N-1
    u(n) = -Kd*Xe(:,n);
    Xe(:,n+1) = (Ad-Lpd*C)*Xe(:,n) + Bd*u(n) + Lpd*C*X(:,n);
    X(:,n+1) = Ad*X(:,n) + Bd*u(n);
end
t = T*[0:N-1];
figure(1); clf; subplot(211)
plot(t,X,'.-'); subplot(212)
plot(t,X-Xe,'.-');

```

mismatch between states (x)
 estimate (\hat{x}) initially.
 $U_k = -K\hat{X}_k$ ← uses estimate.
 estimate update
 true state

Separation Principle : Set estimator poles 2x to 4x faster

Example:

Below, the x axis for the STATES is scaled to be 2x (on the top subplot) what the x axis for the ERROR is, on the lower subplot.

The lower plot show a response due to ONLY the (faster) estimator poles... $\text{eig}(A\text{-LC})$ while the upper plot state responses are due to all 4 poles: $\text{eig}(A\text{-BK})$ and $\text{eig}(A\text{-LC})$

Because $\text{eig}(A\text{-LC})$ poles are FASTER (by design), the upper subplot is (by design) dominated by the slower (state feedback controller) poles, $\text{eig}(A\text{-BK})$.

