Today: Reduced-Order Estimators

Lecture 15

ECE 147B
Digital Control

Reading: FPW 8.2.5, 8.3
or: FV 9.5.2, 9.6 (RS-13)

Topics:
- Separation Principle
- Guidelines for pole placement (control and est.)
- Reduced-Order Estimator
- Frequency Response Methods

Next Class: Frequency Response Methods.
- FPW 7.4 <or> FV (6.4), 6.5, (6.6) (RS-1)

Homework 6: Due Tues (in class).

Using the state estimate for feedback...

Recall STATE FEEDBACK
Control uses the true state:

$$\boxed{u(k) = -Kx(k)}$$

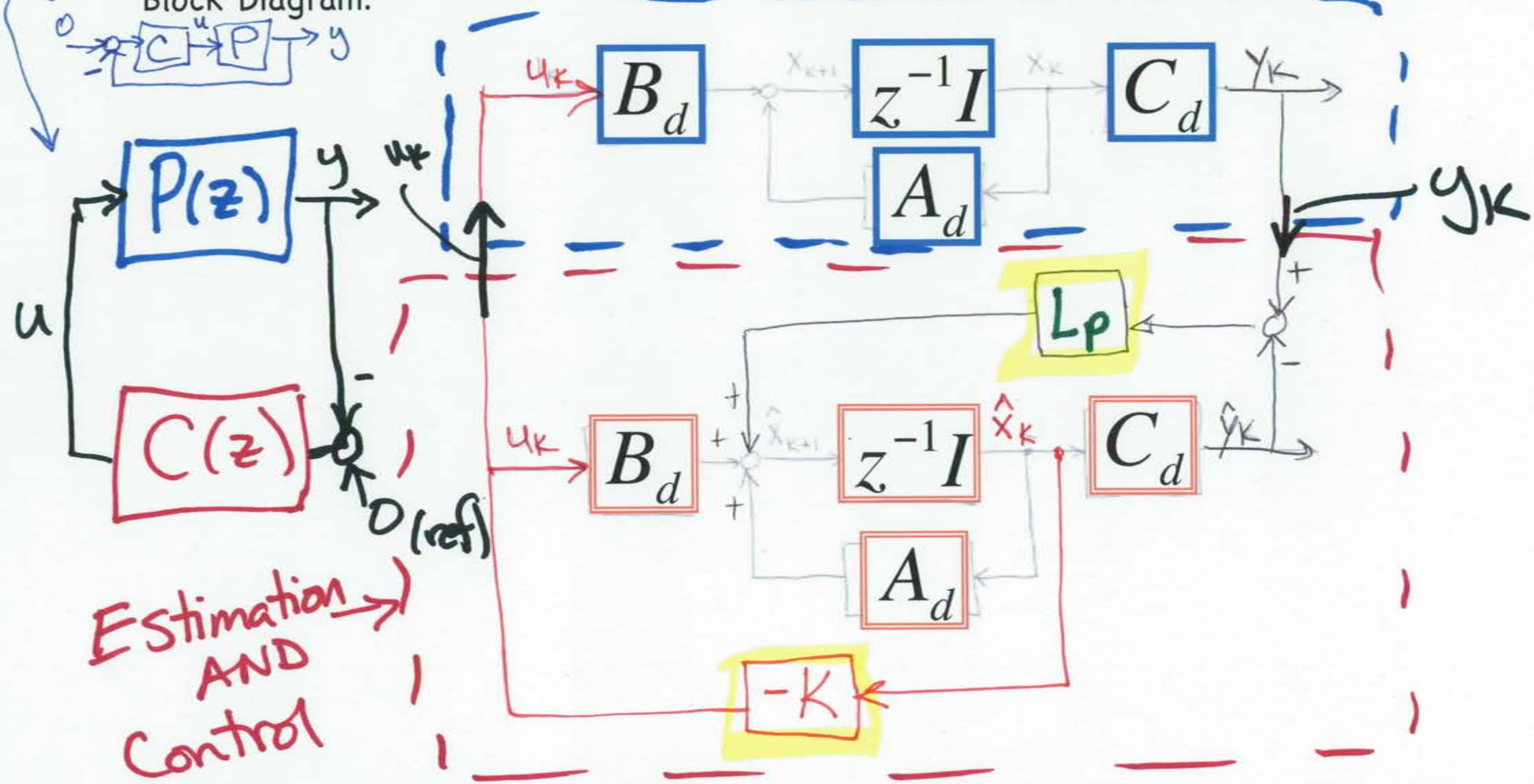
What if we use the state ESTIMATE
for feedback control, instead?

estimate

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

SISO ANALOGY

Block Diagram:



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

MATRIX form for DYNAMICS: for STATES & ERRORS in states

$$\begin{bmatrix} x_{k+1} \\ e_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} (A-BK) & BK \\ \underset{(n\times n)}{0} & (A-LC) \end{bmatrix}}_{\text{"M"}} \begin{bmatrix} x_k \\ e_k \end{bmatrix}$$

$(e \equiv x - \hat{x})$

eigenvalues?

$$\therefore \left(\mathrm{eig}((A-BK)), \mathrm{eig}(A-LC)\right)$$

2n poles, determined "separately"

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

n=1 (state)

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

det(zI - M), M is $\boxed{}$

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

$\boxed{\begin{matrix} z=a \\ z=b \end{matrix}}$ ← poles

n=2

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

"C" (nxn)

"D" = 0 (nxn)

"A" (nxn)

"B" (nxn)

eig(M) → det(zI - M) = 0

↓

are the SAME as:

(eig("A"), eig("B"))

(A & B as defined at left!)

does NOT depend of "C", b/c "D" = 0

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]
```

– symbolic toolbox (in MATLAB)

$$\det\left(\begin{bmatrix} z & 0 & 0 & 0 \\ 0 & z & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & z \end{bmatrix} - \underbrace{\begin{bmatrix} a_1 & a_2 & c_1 & c_2 \\ a_3 & a_4 & c_3 & c_4 \\ 0 & 0 & b_1 & b_2 \\ 0 & 0 & b_3 & b_4 \end{bmatrix}}_{(2n \times 2n)}\right) = \det(zI - M) = 0$$

$A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$ etc.

(2n x 2n) w/ lower-left (n x n) part ALL ZEROS

```
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)
```

= 0 (to find POLES)

Solve for: this part = 0 & this part = 0, separately.

Separation Principle

For feedback with an ESTIMATOR,

$$\hat{x}_{k+1} = \underbrace{A\hat{x}_k + Bu_k}_{\text{"open-loop" dynamics}} + \underbrace{L_p(y_k - C\hat{x}_k)}_{\text{"feedback" part}}$$

$$u_k = -K\hat{x}_k \leftarrow \text{uses "}\hat{x}_k\text{", NOT "}x_k\text{".}$$

1) CONTROL for nth-order system has n poles, determined by eig(A–BK).

Pick (dominant) closed-loop poles, e.g., via "pole placement", for "$\underbrace{\text{eig(A-BK)}}$" ← poles

2) ESTIMATION of n states results in another n poles, determined by $\boxed{\text{eig(A–LC)}}$ ← est'r poles.

Then, pick estimator poles (NOT dominant), Generally $\boxed{\text{(about) 2x - 4x}}$ "faster" than controller poles

3) TOTAL system now has 2*n poles:

& they are the union of $\boxed{\begin{array}{l}\text{eig}(A-BK) \\ \text{and} \\ \text{eig}(A-L_pC)\end{array}}$

i.e., these are 2 SEPARATE (uncoupled) problems!

How to place poles, to set "K" and $\boxed{L_p}$"?

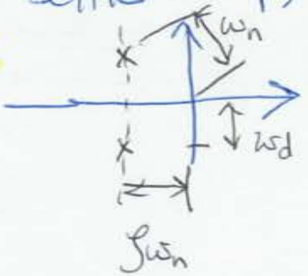
$\therefore$ "2x–4x faster than eig(A–BK)"

eig(A–LC)

Controller poles:

① Use classical control design specifications: (ω_d, t_{rise}, t_{settle}, M_p, etc...) to pick "good" C.L. poles

S-plane



$t_p = \frac{\pi}{\omega_d}$

$M_p = e^{-t_p/\tau}$

$\sigma = \zeta\omega_n$

"$\tau = \frac{1}{\sigma}$"

Z-plane

$$\boxed{\text{"}z = e^{sT}\text{"}}$$

Then use "place" or "acker"...

Estimator poles:

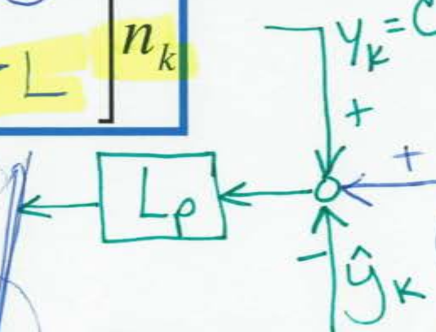
② – Faster than control poles, so controlled CL poles dominate

→ error dynamics "see" only (A–LC) poles, BUT state dynamics (x) "see" both (A–BK) & (A–LC)

Effects of Noise?

"$Ly(k)$" is INSTEAD "$L(y(k)+n(k))$"

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



$\hookrightarrow e_{k+1} = (A-LC)e_k \underset{?}{\pm L} n_k$

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

$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k + n_k - \hat{y}_k)$

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

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

$$\boxed{e_{k+1} = Ae_k - LCe_k - Ln_k} = \boxed{(A-LC)e_k - Ln_k}$$

$\boxed{\therefore \text{Making est. poles TOO FAST makes L LARGE} \Rightarrow \text{sensitive to noise!}}$

Example: control and estimation, w/ MATLAB

CT dynamics

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

Control canonical form!

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

Negative of THIS we want to equal THAT

Desired closed-loop poles

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

WANTED: $\boxed{s^2+20s+200=0}$

$K_1+1=20$

$\boxed{K_1=19}$

$\boxed{K_2=199}$

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

DT (discrete time), [T = 0.01 sec] – system resulting from ZOH

```
M=expm([A, B; zeros(m,n+m)]*T)
Ad = M(1:n, 1:n);
Bd = M(1:n, n+1:end);
```

"$z = e^{sT}$"

```
pdt = exp(pct*T);
```

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

"Trick" to get A_d & B_d for ZOH (plus sampling) DT system

OR, use "c2d"

$K = [K_1, K_2]$

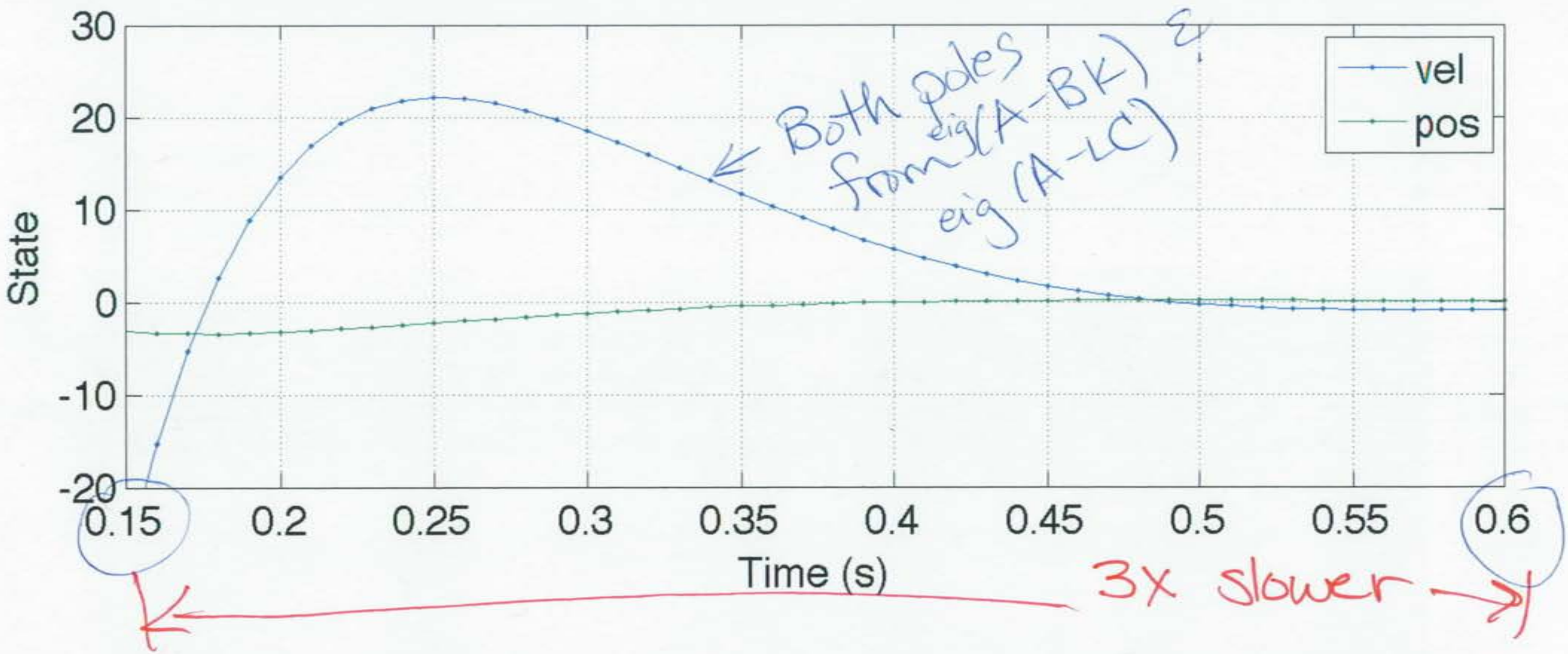
$\boxed{K = [19, 199]}$

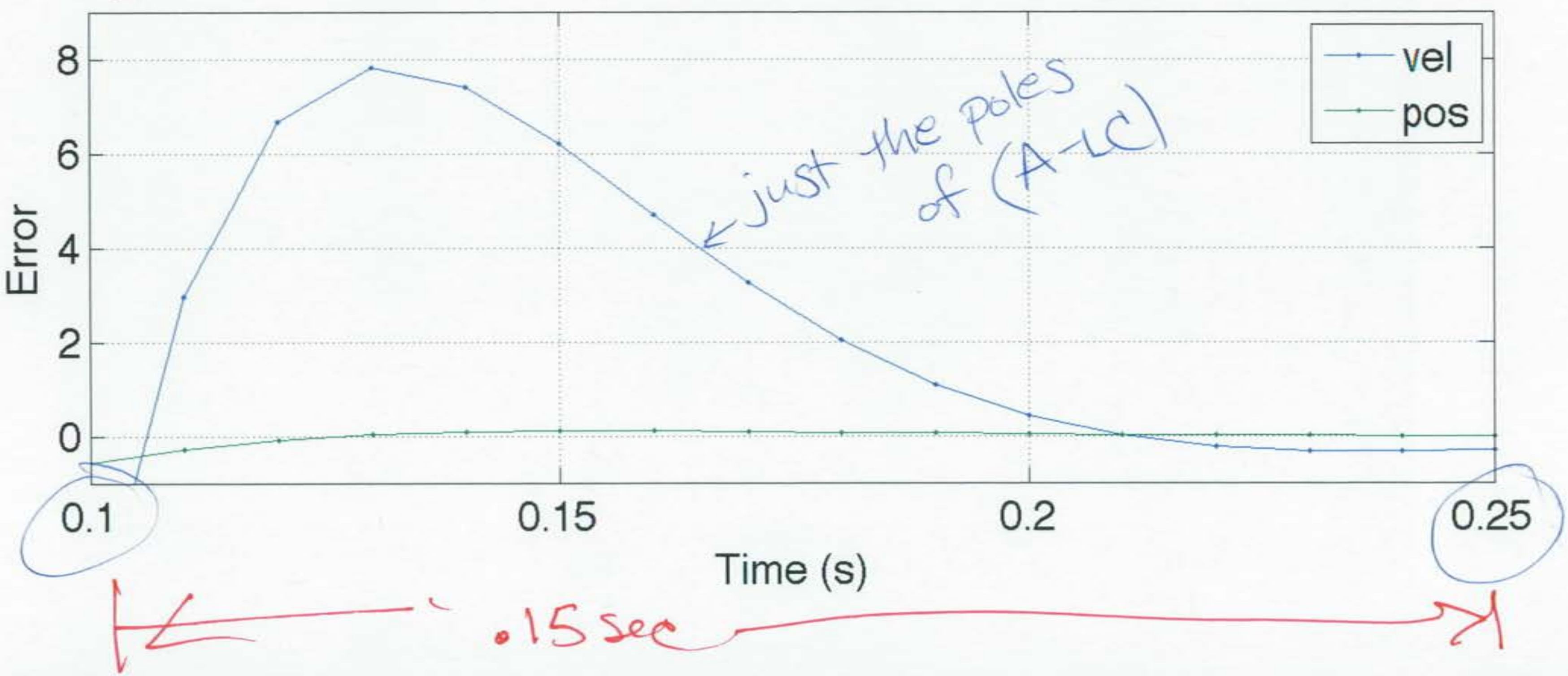
CT dynamics

```
Kd = place(Ad, Bd, pdt)
```

$\boxed{K_d = [18.12, 180.88]}$

Not QUITE the same, but close.


State
Time (s)
vel
pos
Both poles from eig(A-BK) & eig(A-LC)
3x slower


Error
Time (s)
vel
pos
just the poles of (A-LC)
.15 sec

Reduced-order Estimator

What if you only want to "estimate" a subset of the entire state vector? e.g., You only want to estimate "unmeasured" states, which are not inside y. For example, in LAB:

Position: encoder - measured, x

Velocity: Not measured! ← unmeasured, $\dot{x}$

Let's partition the states, $x(k) = \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix}$ a: directly measured

b: not measured ← only estimate $\hat{x}_b$ w/ red-ord. est.

we will get this directly from measurement

$$\underbrace{\begin{bmatrix} x_a(k+1) \\ x_b(k+1) \end{bmatrix}}_{\text{"}x_{k+1}\text{"}} = \underbrace{\begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix}}_{A} \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix} + \underbrace{\begin{bmatrix} B_a \\ B_b \end{bmatrix}}_{B} u(k)$$

$$\boxed{\#1}\quad \underbrace{x_b(k+1)}_{\text{unknown}} = \underbrace{A_{bb}x_b(k)}_{\text{unknown}} + \underbrace{\left(A_{ba}x_a(k) + B_b u(k)\right)}_{\text{Known}}$$

Error dynamics

$\boxed{\#2}$ → write an eqn for $\hat{x}_b(k+1)$ ... & then $\boxed{\#3}$ is calculate by $\boxed{\#1}$ minus $\boxed{\#2}$

Reduced-order Estimator

from last page

$$* \quad x_a(k+1) = A_{aa}x_a(k) + \boxed{A_{ab}x_b(k)} + B_a u(k)$$

Think of this as "Cx" in "$y=Cx$"...

This looks a lot like

$$\boxed{\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + \cancel{Du(k)} \end{aligned}}$$

$\rightarrow x_b(k+1) = A_{bb}x_b(k) + (A_{ba}x_a(k) + B_b u(k))$ ① ② ③

$\rightarrow (x_a(k+1) - A_{aa}x_a(k) - B_a u(k)) = A_{ab}x_b(k)$

④ "$y(k)$" ⑤ $= Cx(k)$

...if we make the following SUBSTITUTIONS:

Was	→	Now	
1) ① $x(k)$	→	$x_b(k)$	← states TO BE ESTIMATED
2) ② A	→	A_{bb}	← a sub-part of the full "A" matrix.
3) ③ $Bu(k)$	→	$(A_{ba}x_a(k) + B_b u(k))$	← "Known inputs now include BOTH "Bu" and also terms depending on $x_a(k)$ [known/measured].
4) ④ $y(k)$	→	$(x_a(k+1) - A_{aa}x_a(k) - B_a u(k))$	← What is left on LHS of * once "$A_{ab}x_b(k)$" (like "Cx") is moved to RHS.
5) ⑤ C	→	A_{ab}	← "$A_{ab}x_b(k)$" is now analogous to "Cx"

GOAL is to match THIS form:
(But estimate ONLY $x_b(k)$...)

$$\boxed{\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - C\hat{x}(k))}$$

Use piece ① through ⑤ above!

Reduced-order Estimator

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - C\hat{x}(k))$$ ← Traditional (not "reduced") form, which we wish to match.

$\boxed{\#2}$ $$\hat{x}_b(k+1) = A_{bb}\hat{x}_b(k) + (A_{ba}x_a(k) + B_b u(k)) + L_r(\underbrace{[x_a(k+1) - A_{aa}x_a(k) - B_a u(k)]}_{=A_{ab}x_b(k)} - A_{ab}\hat{x}_b(k))$$

Uh oh!

from eqn $\boxed{*}$, last page...

We have a new estimator equation, as desired, but there is a PROBLEM! We do not have access to "$x_a(k+1)$" yet! (Causality...)
(To be discussed NEXT PAGE...)

Recall:

$\boxed{\#1}$ $$x_b(k+1) = A_{bb}x_b(k) + (A_{ba}x_a(k) + B_b u(k))$$

So, use "place" with
"A" = A_{bb}
"C" = A_{ab}

Error Dynamics? $e_{xb} \equiv x_b - \hat{x}_b$, So, take $\boxed{\#1}$ minus $\boxed{\#2}$

$$x_b(k+1) - \hat{x}_b(k+1) = A_{bb}(x_b(k) - \hat{x}_b(k)) - L_r(A_{ab}x_b(k) - A_{ab}\hat{x}_b(k))$$

$$\therefore \boxed{e_b(k+1) = [\underset{\text{"A"}}{A_{bb}} - \underset{\text{"LC"}}{L_r A_{ab}}]\, e_b(k)}$$ ← poles set by $\boxed{\text{"}A_{bb} - L_r A_{ab}\text{"}}$

Reduced-order Estimator

* <u>Issue</u>: How can we include "$X_a(k+1)$" in our actual <u>implementation</u> of the reduced-order estimator?

<u>Ans</u>: Let $\boxed{X_c(k) \equiv \hat{X}_b(k) - L_r X_a(k)}$

Get $\hat{X}_b(k+1)$ from $\boxed{\#2}$

And note, $X_a(k+1) = A_{aa} X_a(k) + A_{ab} X_b(k) + B_a u(k)$

$$\therefore X_c(k+1) = \hat{X}_b(k+1) - L_r X_a(k+1)$$

Cancels out!

$$= \left(A_{bb} \boxed{\hat{X}_b(k)} + (A_{ba} X_a(k) + B_b \boxed{u(k)}\right) + L_r\left(\cancel{A_{ab} X_b(k)} - A_{ab} \boxed{\hat{X}_b(k)}\right) -$$

$$L_r\left(A_{aa} X_a(k) + \cancel{A_{ab} X_b(k)} + B_a \boxed{u(k)}\right)$$

Regrouping:

$$\boxed{X_c(k+1) = (A_{bb} - L_r A_{ab}) \hat{X}_b(k) + (A_{ba} - L_r A_{aa}) X_a(k) + (B_b - L_r B_a) u(k)}$$

And then, $\hat{X}_b(k) = X_c(k) + L_r X_a(k)$

$$u(k) = -K \begin{bmatrix} x_a(k) \\ x_b(k) \end{bmatrix}$$