OUTPUT-FEEDBACK CONTROL

7.1: Open-loop and closed-loop estimators

Open-loop estimators

- State feedback is impractical since we don’t know the state!
- But, what if we can estimate the state?

**IDEA:** Since we know system dynamics, simulate system in real-time.

- If \( x(t) \) is the true state, \( \hat{x}(t) \) is called the state estimate.
- We want \( \hat{x}(t) = x(t) \), or at least \( \hat{x}(t) \to x(t) \). How do we build this?
- To start, use our knowledge of the plant

\[
\dot{x}(t) = Ax(t) + Bu(t).
\]

- Let our state estimate be

\[
\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t).
\]

- This is called an “open-loop estimator.”

Some troubling issues:

- We need our model to be very accurate!
● What do we use for \( \hat{x}(0) \)?

● What does disturbance do?

■ Let’s analyze our open-loop estimator by examining the state-estimate error

\[
\tilde{x}(t) = x(t) - \hat{x}(t).
\]

■ We want \( \tilde{x}(t) = 0 \). For our estimator,

\[
\dot{\tilde{x}}(t) = \dot{x}(t) - \dot{\hat{x}}(t)
\]

\[
= Ax(t) + Bu(t) - A\hat{x}(t) - Bu(t)
\]

\[
= A\tilde{x}(t).
\]

■ So,

\[ \tilde{x}(t) = e^{At}\tilde{x}(0). \]

■ Hence, \( \hat{x}(t) \rightarrow x(t) \) if \( A \) is stable!

● (This is not too impressive though since \( \hat{x}(t) \rightarrow x(t) \) because both \( \hat{x}(t) \) and \( x(t) \) go to zero).

■ We need to improve our estimator:

● Speed up convergence.

● Reduce sensitivity to model uncertainties.

● Counteract disturbances.

● Have convergence even when \( A \) is unstable.

■ Key Point: Use feedback of measured output.
This is called a “closed-loop estimator” (assuming $\hat{y} = C \hat{x}$).

Note: If $L = 0$ we have an open-loop estimator.

$$\dot{\hat{x}}(t) = A \hat{x}(t) + Bu(t) + L (y(t) - \hat{y}(t)).$$

Let’s look at the error.

$$\hat{x}(t) = \dot{x}(t) - \hat{x}(t)$$

$$= Ax(t) + Bu(t) - A \hat{x}(t) - Bu(t) - L (y(t) - C \hat{x}(t))$$

$$= A\hat{x}(t) - L (Cx(t) - C \hat{x}(t))$$

$$= (A - LC) \hat{x}(t);$$

$$\hat{x}(t) = e^{(A-LC)t} \hat{x}(0),$$

or, $\hat{x}(t) \rightarrow x(t)$ if $A - LC$ is stable, for any value of $\hat{x}(0)$ and any $u(t)$, whether or not $A$ is stable.

In fact, we can look at the dynamics of the state estimate error to quantitatively evaluate how $\hat{x}(t) \rightarrow x(t)$.

$$\dot{\hat{x}}(t) = (A - LC) \hat{x}(t)$$

has dynamics related to the roots of the characteristic equation.
\[ \chi_{ob}(s) = \det(sI - A + LC) = 0. \]

- So, for our estimator, we specify the convergence rate of \( \hat{x}(t) \rightarrow x(t) \) by choosing desired pole locations: Choose \( L \) such that

\[ \chi_{ob,des}(s) = \det(sI - A + LC). \]

- This is called the “observer gain problem”.

- In Simulink, the following diagram implements a closed-loop estimator. The output is \( \hat{x} \).

![Simulink diagram](image)

**Integration dynamics**

- It can be helpful to add integration dynamics to the observer to ensure that steady-state errors go to zero.

- Instead of comparing \( \hat{y}(t) \) to \( y(t) \), we compare the integral of \( \hat{y}(t) \) to the integral of \( y(t) \).

- To do so, we define a new “state” that integrates the output

\[ x_i(t) = \int_0^t y(\tau) \, d\tau \quad \text{or} \quad \dot{x}_i(t) = y(t) = Cx(t) + Du(t). \]
Then, we can combine the true system dynamics with the integration dynamics

\[
\begin{bmatrix}
\dot{x}_i(t) \\
\dot{x}(t) \\
\dot{x}_a(t)
\end{bmatrix} = \begin{bmatrix} 0 & C & \cdot \\
0 & A & \cdot \\
A_a & x_a(t)
\end{bmatrix} \begin{bmatrix} x_i(t) \\
x(t) \\
x_a(t)
\end{bmatrix} + \begin{bmatrix} D \\
B \\
B_a
\end{bmatrix} u(t)
\]

\[
y(t) = \begin{bmatrix} I & 0 \\
C_a & \cdot \\
\cdot & \cdot
\end{bmatrix} \begin{bmatrix} x_i(t) \\
x(t) \\
x_a(t)
\end{bmatrix}.
\]

The estimator then is designed such that

\[
\dot{x}_a(t) = A_a \dot{x}_a(t) + B_a u(t) + L_a (x_i(t) - \dot{x}_i(t)) = A_a \dot{x}_a(t) + B_a u(t) + L_a C_a (x_a(t) - \dot{x}_a(t)).
\]

The estimation error has dynamics

\[
\dot{x}(t) = (A_a x_a(t) + B_a u(t)) - (A_a \dot{x}_a(t) + B_a u(t) + L_a C_a \dot{x}_a(t)) = (A_a - L_a C_a) \dot{x}_a(t).
\]

Therefore, we need to design an 

\[
L_a \text{ matrix such that } (A_a - L_a C_a) \text{ has nice properties.}
\]

As a check, is the augmented system even observable?

Notice that the new observability matrix is

\[
O_a = \begin{bmatrix}
C_a \\
C_a A_a \\
C_a A_a^2 \\
\vdots
\end{bmatrix} = \begin{bmatrix} I & 0 \\
0 & C \\
0 & CA \\
\vdots & \vdots
\end{bmatrix} = \begin{bmatrix} I & 0 \\
0 & O
\end{bmatrix}.
\]

So, if the original system is observable, the integrator-augmented system is also observable.
7.2: The observer gain design problem

- We would like a method for computing the observer gain vector $L$ given a set of desired closed-loop observer gains $\chi_{ob,des}(s)$.

**Bass–Gura inspired method**

- We want a specific characteristic equation for $A - LC$.
- Suppose $\{C, A\}$ is observable. Put $\{A, B, C, D\}$ in observer canonical form.
- That is, find $T$ such that
  \[
  T^{-1}AT = A_o = \begin{bmatrix}
  -a_1 & 1 & 0 \\
  -a_2 & \cdots & 1 \\
  \vdots & \ddots & \vdots \\
  -a_n & 0 & \cdots & 0
  \end{bmatrix}
  \quad \text{and} \quad
  CT = C_o = \begin{bmatrix}
  1 & 0 & \cdots & 0
  \end{bmatrix}
  \]
  where $\det(sI - A) = s^n + a_1 s^{n-1} + \cdots + a_n$.
- Apply feedback $L_o$ to the observer realization to end up with $A_o - L_o C_o$.
- Note, $L_o = \begin{bmatrix} l_1 & \cdots & l_n \end{bmatrix}^T$, so

  \[
  L_o C_o = \begin{bmatrix}
  l_1 & 0 & \cdots & 0 \\
  l_2 & \cdots & \vdots & \ddots \\
  l_n & 0 & \cdots & 0
  \end{bmatrix}
  .
  \]

- Useful because characteristic equation obvious.
\[
A_o - L_o C_o = \begin{bmatrix}
-(a_1 + l_1) & 1 & 0 \\
-(a_2 + l_2) & \ddots & \vdots \\
\vdots & & 1 \\
-(a_n + l_n) & 0 & \cdots & 0
\end{bmatrix},
\]

still in observer form!

- After feedback with \( L_o \) the characteristic equation is
  \[
  \det(sI - A_o + L_o C_o) = s^n + (a_1 + l_1)s^{n-1} + \cdots + (a_n + l_n).
  \]

- If we set \( l_1 = \alpha_1 - a_1, \ldots, l_n = \alpha_n - a_n \) then we get the desired characteristic polynomial.

- Now, we transform back to the original realization

  \[
  \det(sI - A_o + L_o C_o) = \det(sI - TA_o T^{-1} + TL_o C o T^{-1})
  = \det(sI - A + TL_o C).
  \]

- So, if we use feedback

  \[
  L = TL_o
  = T \left[ (\alpha_1 - a_1) \cdots (\alpha_n - a_n) \right]^T
  \]

  we will have the desired characteristic polynomial.

- One remaining question: What is \( T \)? We know \( T = O^{-1} O_o \) and

  \[
  O_o = \begin{bmatrix}
  1 & 0 & \cdots & 0 \\
  a_1 & 1 & & \\
  \vdots & \ddots & \vdots & 0 \\
  a_{n-1} & \cdots & a_1 & 1
  \end{bmatrix}^{-1}
  \]
So,

\[ L = O^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & a_1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} (\alpha_1 - a_1) \\ (\alpha_2 - a_2) \\ \vdots \\ (\alpha_n - a_n) \end{bmatrix} \]

This is the **Bass–Gura formula** for \( L \).

**Duality!**

This is another example of duality. Notice that if you substitute \( A \leftarrow A^T, B \leftarrow C^T, L \leftarrow K^T \) in the Bass–Gura controller design procedure, you will get exactly this equation. Why?

Notice that we wish to design certain pole locations:

\[ \text{eig}(A - LC) = \text{eig}(A^T - C^T L^T) \]

which has the same form as the controller design problem

\[ \text{eig}(A - BK) \]

if we make the above substitutions.

**The Ackermann-inspired method**

The observer-gain problem is “dual” to the controller-gain problem. Replace a controller-design procedure’s inputs \( A \leftarrow A^T, B \leftarrow C^T, K \leftarrow L^T \). Design the controller. Then, \( L \) will be the observer gains.

Recall, \( K = \begin{bmatrix} 0 & \cdots & 1 \end{bmatrix} C(A, B)^{-1} \chi_d(A) \).

By duality,

\[ L = \left[ \begin{bmatrix} 0 & \cdots & 1 \end{bmatrix} C(A, B)^{-1} \chi_d(A) \right]^T_{A \leftarrow A^T, B \leftarrow C^T}. \]
\[
\chi_d(T) \left[ C^T A^T C^T \cdots A^{n-1}C^T \right]^{-T} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}
\]

\[
= \chi_d(A) \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}
\]

\[
= \chi_d(A) \mathcal{O}(C, A)^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.
\]

- In MATLAB,

```matlab
L = acker(A', C', poles)';
L = place(A', C', poles)';
```
7.3: Discrete-time prediction estimator

- In discrete-time, we can do the same thing. The picture looks like

![Discrete-time prediction estimator diagram]

- The update equation for the closed-loop (prediction) estimator is

\[ \hat{x}_p[k + 1] = A\hat{x}_p[k] + Bu[k] + L_p \left( y[k] - C\hat{x}_p[k] \right). \]

- The prediction estimation error can likewise be written as

\[ \ddot{x}[k + 1] = (A - L_pC) \ddot{x}[k], \]

which has dynamics related to the roots of the characteristic equation

\[ \chi_{ob}(z) = \det(zI - A + L_pC) = 0. \]

- We specify the convergence rate of \( \hat{x}_p[k] \rightarrow x[k] \) by choosing desired pole locations: Choose \( L_p \) such that

\[ \chi_{ob,des}(z) = \det(zI - A + L_pC). \]

**EXAMPLE:** Let \( G(s) = \frac{1}{s^2} \) and measure \( y[k] \). Let

\[ x[k] = \begin{bmatrix} y[k] \\ \dot{y}[k] \end{bmatrix}, \]

then

\[ A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \]
We desire $\tilde{x}[k]$ to decay with poles $z_p = 0.8 \pm 0.2j$, or

$$\chi_{obs, des}(z) = z^2 - 1.6z + 0.68.$$  

$$\det(zI - A + L_pC) = \det \begin{bmatrix} z - 1 + l_1 & -T \\ l_2 & z - 1 \end{bmatrix} = z^2 + z(l_1 - 2) + l_2T - l_1 + 1.$$  

So,

$$l_1 - 2 = -1.6$$  

$$l_2T - l_1 + 1 = 0.68$$

or

$$L_p = \begin{bmatrix} 0.4 \\ 0.08/T \end{bmatrix}.$$  

The estimator is

$$\hat{x}_p[k + 1] = A\hat{x}_p[k] + Bu[k] + L_p(y[k] - C\hat{x}_p[k])$$  

$$= (A - L_pC)\hat{x}_p[k] + Bu[k] + L_py[k],$$

or

$$\begin{bmatrix} \hat{y}_p[k + 1] \\ \hat{y}_p[k + 1] \end{bmatrix} = \begin{bmatrix} 0.6 & T \\ -0.08/T & 1 \end{bmatrix} \begin{bmatrix} \hat{y}_p[k] \\ \hat{y}_p[k] \end{bmatrix} + \begin{bmatrix} T^2 \\ 2/T \end{bmatrix} u[k] + \begin{bmatrix} 0.4 \\ 0.08/T \end{bmatrix} y[k].$$

In general, we can arbitrarily select the prediction estimator poles iff $\{C, A\}$ is observable.

The observer-gain problem is "dual" to the controller-gain problem. Replace a controller-design procedure’s inputs $A \leftarrow A^T, B \leftarrow C^T, K \leftarrow L_p^T$. Design the controller. Then, $L_p$ will be the observer gains.
7.4: Compensator design: Separation principle

- Now that we have a structure to estimate the state $x(t)$, let’s feed back $\hat{x}(t)$ to control the plant. That is,

$$u(t) = r(t) - K\hat{x}(t),$$

where $K$ was designed assuming that $u(t) = r(t) - Kx(t)$. Is this going to work? How risky is it to interconnect two well-behaved, stable systems? (Assume $r(t) = 0$ for now).

What is inside the dotted line is equivalent to our classical design compensator:

$$\hat{x}(t) = A\hat{x}(t) + Bu(t) + L(y(t) - \hat{y}(t))$$

where $G(s)$ is described by

$$\hat{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t),$$

and $D(s)$ is described by
\[
\dot{x}(t) = A\hat{x}(t) + Bu(t) + L(y(t) - \hat{y}(t))
\]

\[
u(t) = -K\hat{x}(t),
\]

so that we have

\[
\dot{x}(t) = Ax(t) - BK\hat{x}(t)
\]

\[
\dot{\hat{x}}(t) = (A - BK)\hat{x}(t) + L(Cx(t) + Du(t) - C\hat{x}(t) - Du(t))
\]

\[
= (A - BK - LC)\hat{x}(t) + LCx(t).
\]

- Our combined closed-loop-system state equations are

\[
\begin{bmatrix}
\dot{x}(t) \\
\dot{\hat{x}}(t)
\end{bmatrix}
= \begin{bmatrix}
A & -BK \\
LC & A - BK - LC
\end{bmatrix}
\begin{bmatrix}
x(t) \\
\hat{x}(t)
\end{bmatrix},
\]

or, in terms of \(\tilde{x}(t)\),

\[
\dot{\tilde{x}}(t) = \dot{x}(t) - \dot{\hat{x}}(t)
\]

\[
= Ax(t) - BK\hat{x}(t) - LCx(t) - (A - BK - LC)\hat{x}(t)
\]

\[
= (A - LC)\tilde{x}(t)
\]

\[
\dot{x}(t) = Ax(t) - BK(x(t) - \tilde{x}(t)),
\]

or,

\[
\begin{bmatrix}
\dot{x}(t) \\
\dot{\tilde{x}}(t)
\end{bmatrix}
= \begin{bmatrix}
A - BK & BK \\
0 & A - LC
\end{bmatrix}
\begin{bmatrix}
x(t) \\
\tilde{x}(t)
\end{bmatrix}.
\]

- Note, we have simply changed coordinates:

\[
\begin{bmatrix}
x(t) \\
\hat{x}(t)
\end{bmatrix}
= \begin{bmatrix}
I & 0 \\
I & -I
\end{bmatrix}
\begin{bmatrix}
x(t) \\
\tilde{x}(t)
\end{bmatrix}.
\]
With this change of coordinates, we may easily find the closed-loop poles of the combined state-feedback/estimator system: The eigenvalues of a block-upper-triangular $2 \times 2$ matrix are the collection of the eigenvalues of the two diagonal blocks.

Therefore, the $2n$ poles are the $n$ eigenvalues of $A - BK$ combined with the $n$ eigenvalues of $A - LC$.

But, we *designed* the $n$ eigenvalues of $A - BK$ to give good (stable) state-feedback performance, and we *designed* the $n$ eigenvalues of $A - LC$ to give good (stable) estimator performance. Therefore, the closed-loop system is also stable.

This is such an astounding conclusion that it has been given a special name: The “Separation Principle”.

It implies that we can design a compensator in two steps:

1. Design state-feedback assuming the state $x(t)$ is available.
2. Design the estimator to estimate the state as $\hat{x}(t)$.

and, that $u(t) = -K\hat{x}(t)$ works!
7.5: The compensator: Continuous- and discrete-time

The continuous-time compensator

- What is $D(s)$? We know the closed-loop poles, which are the roots of $1 + D(s)G(s) = 0$, and we know the plant’s open-loop poles. What are the dynamics of $D(s)$ itself?

- Start with a state-space representation of $D(s)$

$$
\dot{x}(t) = (A - BK - LC)\dot{x}(t) + Ly(t) - LDu(t)
$$

$$
u(t) = -K\dot{x}(t)
$$

so that

$$
D(s) = \frac{U(s)}{Y(s)} = -K(sI - A + BK + LC - LDK)^{-1}L.
$$

- The poles of $D(s)$ are the roots of $\det(sI - A + BK + LC - LDK) = 0$.

- These are neither the controller poles nor the estimator poles.

- $D(s)$ may be unstable even if the plant is stable and the closed-loop system is stable.

The discrete-time compensator

- The results in discrete-time are essentially the same.

$$
D(z) = \frac{U(z)}{Y(z)} = -K\left(zI - A + BK + L_pC - L_pDK\right)^{-1}L_p.
$$

- The poles of $D(z)$ are the roots of

$$
\det\left(zI - A + BK + L_pC - L_pDK\right) = 0.
$$

- The compensator has the block-diagram:
Note that the control signal $u[k]$ only contains information about $y[k - 1]$, $y[k - 2]$, \ldots, not about $y[k]$.

So, our compensator is not taking advantage of the most current measurement. (More on this later).

```matlab
for loop:
    u=-K*xhatp;
    \% now, wait until sample time...
    A2D(y);
    D2A(u); \% u depends on past y, not current.
    xhatp=(A-B*K-Lp*C+Lp*D*K)*xhatp+Lp*y;
end loop.
```

**EXAMPLE:** $G(s) = \frac{1}{s^2}; T = 1.4$ seconds.

Design a compensator such that response is dominated by the poles $z_p = 0.8 \pm 0.25j$.

System description

$$A = \begin{bmatrix} 1 & 1.4 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1.4 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = [0],$$

Control design: Find $K$ such that $\det(zI - A + BK) = z^2 - 1.6z + 0.7$. This leads to $K = \begin{bmatrix} 0.05 & 0.25 \end{bmatrix}$. 
- Estimator design: Choose poles to be faster than control roots. Let's radially project \(z_p = 0.8 \pm 0.25j\) toward the origin, or \(z_{pe} = 0.8z_p\).

- Find \(L_p\) such that \(\det(zI - A + L_pC) = z^2 - 1.28z + 0.45\). This leads to \(L_p = \begin{bmatrix} 0.72 \\ 0.12 \end{bmatrix}\).

- So, our compensator is
  \[
  \hat{x}_p[k + 1] = \begin{bmatrix} 0.23 & 1.16 \\ -0.19 & 0.65 \end{bmatrix} \hat{x}_p[k] + \begin{bmatrix} 0.72 \\ 0.12 \end{bmatrix} y[k]
  \]
  \[
  u[k] = \begin{bmatrix} -0.05 & -0.25 \end{bmatrix} \hat{x}_p[k],
  \]
  
  or,
  \[
  D(z) = -0.68 \frac{z - 0.87}{z - 0.44 \pm 0.423j} = -0.68 \frac{z - 0.87}{z^2 - 0.88z + 0.374}.
  \]

- Let's see the root locus of \(D(z)G(z)\). (Note, since \(D(z)\) has a negative sign, must use \(0^\circ\) locus).
Using the state representation of the plant and our compensator, we can simulate the closed-loop system to find $x[k]$, $\hat{x}_p[k]$, $\tilde{x}[k]$, and $u[k]$.

**EXAMPLE:** In MATLAB,

```matlab
Q=[A  -B*K; L*C  A-B*K-L*C];
% get resp. of u, x, xhat to init. condition
[u,X]=dinitial(Q,zeros([4,1]),[0 0 -K],0,x0);
% get the estimate error.
xtilde=X(:,1:2)-X(:,3:4);
```

[Graphs showing system and estimator state dynamics, and error dynamics]
7.6: Current estimator/ compensator

- Using the prediction estimator to build our compensator, we found that the control effort $u[k]$ did not utilize the most current measurement $y[k]$, only past values of $y$: $y[k - 1], y[k - 2] \ldots$

**IMPLICATION:** Cannot implement $D(z) = \frac{z - a}{z - b}$ (either lead or lag) to control a system since

$$G(z) = \frac{U(z)}{Y(z)} = 1 + \frac{b - a}{z - b}.$$  

- That is, $u[k] = f(y[k], \ldots)$. To develop the current estimate $\hat{x}_c[k]$, consider “tuning-up” our prediction estimate $\hat{x}_p[k]$ at time $k$ with $y[k]$.

\[
\hat{x}_c[k], \hat{x}_p[x]
\]

\[
\hat{x}_p[k] : \text{Estimate just before measurement at } k
\]

\[
\hat{x}_c[k] : \text{Estimate just after measurement at } k.
\]

**IMPLEMENTATION:**

- “Time update”: Predict new state from old estimate, system dynamics
  $$\hat{x}_p[k] = A\hat{x}_c[k - 1] + Bu[k - 1].$$

- “Measurement update”: Measure output, use it to update the estimate
  $$\hat{x}_c[k] = \hat{x}_p[k] + L_c \left( y[k] - C \hat{x}_p[k] \right).$$
• $L_c$ is called the “current estimator gain.”

■ Questions: How are $\hat{x}_c[k]$ and $\hat{x}_p[k]$ and how are $L_c$ and $L_p$ related?

■ What is the $\hat{x}_c[k]$ recursion relation?

\[
\hat{x}_c[k + 1] = \hat{x}_p[k + 1] + L_c \left( y[k + 1] - C \hat{x}_p[k + 1] \right) \\
= (I - L_c C) \hat{x}_p[k + 1] + L_c y[k + 1] \\
= (I - L_c C) (A\hat{x}_c[k] + Bu[k]) + L_c y[k + 1] \\
= (A - L_c CA) \hat{x}_c[k] + (B - L_c CB) u[k] + L_c y[k + 1].
\]

■ So,

\[
\hat{x}_c[k] = f(\hat{x}_c[k - 1], u[k - 1], y[k]).
\]

■ What about the estimate error?

\[
\tilde{x}[k + 1] = x[k + 1] - \hat{x}_c[k + 1] \\
= x[k + 1] - (\hat{x}_p[k + 1] + L_c C x[k + 1] - L_c C \hat{x}_p[k + 1]) \\
= (I - L_c C) (x[k + 1] - \hat{x}_p[k + 1]) \\
= (I - L_c C) (Ax[k] + Bu[k] - A\hat{x}_c[k] - Bu[k]) \\
= (A - L_c C \underbrace{A}_{\text{new!}}) \tilde{x}[k].
\]

■ So, the current estimator error has dynamics related to the roots of

\[
\det (zI - A + L_c CA) = 0.
\]

■ What about the $\hat{x}_p$ recursion relation?

\[
\hat{x}_p[k + 1] = A\hat{x}_c[k] + Bu[k]
\]
\[
= A \left( \hat{x}_p[k] + L_c \left( y[k] - C \hat{x}_p[k] \right) \right) + Bu[k] \\
= A\hat{x}_p[k] + Bu[k] + AL_c \left( y[k] - C \hat{x}_p[k] \right).
\]

- Compare this equation with the prediction estimate recursive equation. You will notice that is the same except that

\[
L_p = AL_c.
\]

- So, \( \hat{x}_p \) in the current-estimator equations is the same quantity \( \hat{x}_p \) in the prediction-estimator equations.

- This implies that if we define \( \tilde{x} = x - \hat{x}_p \) (not \( x - \hat{x}_c \)), then

\[
\tilde{x}[k+1] = (A - L_p C) \tilde{x}[k] = (A - AL_c C) \tilde{x}[k].
\]

- So, in summary

\[
\tilde{x} = x - \hat{x}_p \quad \Rightarrow \quad \tilde{x}[k+1] = (A - L_c CA) \tilde{x}[k] \\
\tilde{x} = x - \hat{x}_c \quad \Rightarrow \quad \tilde{x}[k+1] = (A - AL_c C) \tilde{x}[k].
\]

- These estimate errors have the same poles. They represent the dynamics of the block diagrams:
**Design of \( L_c \)**

1. Relate coefficients of

\[
\det(zI - A + L_c CA) = \chi_{ob,des}(z).
\]

2. Ackermann’s formula for \( L_c \)

\[
L_p = \chi_{ob,des}(z)O^{-1}\{C, A\} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}
\]

replace \( C \leftarrow CA \) to find \( L_c \)

\[
L_c = \chi_{ob,des}(A)O^{-1}\{CA, A\} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} = \chi_{ob,des}(A) \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^n \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.
\]

- In MATLAB,

\[
Lc=acker(A', (C*A)', poles)';
Lc=place(A', (C*A)', poles)';
\]

3. Find \( L_p \) and then \( L_c = A^{-1}L_p \).
7.7: Compensator design using the current estimator

- **Plant equations**

\[ x[k + 1] = Ax[k] + Bu[k] \]
\[ y[k] = Cx[k]. \]

- **Estimator equations**

\[ \hat{x}_p[k + 1] = A\hat{x}_c[k] + Bu[k] \]
\[ \hat{x}_c[k] = \hat{x}_p[k] + L_c (y[k] - C\hat{x}_p[k]). \]

- **Control**

\[ u[k] = -K\hat{x}_c[k]. \]

- Therefore, we have

\[ x[k + 1] = Ax[k] - BK\hat{x}_c[k] \]
\[ \hat{x}_c[k + 1] = (I - L_c C) A\hat{x}_c[k] + (I - L_c C) Bu[k] + L_c y[k + 1] \]
\[ = (I - L_c C) (A - BK)\hat{x}_c[k] + L_c y[k + 1]. \]

- With \( y[k + 1] = CAx[k] + CBu[k], \) then

\[ \hat{x}_c[k + 1] = (A - BK - L_c CA)\hat{x}_c[k] + L_c CAx[k]. \]

- **Our \( 2n \)-order, closed-loop system is**

\[
\begin{bmatrix}
  x[k + 1] \\
  \hat{x}_c[k + 1]
\end{bmatrix}
= \begin{bmatrix}
  A & -BK \\
  L_c CA & A - BK - L_c CA
\end{bmatrix}
\begin{bmatrix}
  x[k] \\
  \hat{x}_c[k]
\end{bmatrix}.
\]

- Compare this to the prediction estimator feedback case:

\[ L_c \leftarrow L_p \]
\[ CA \leftarrow C \]
In terms of $$\tilde{x}[k + 1] = x[k + 1] - \hat{x}_c[k + 1]$$

$$
\begin{bmatrix}
  x[k + 1] \\
  \tilde{x}[k + 1]
\end{bmatrix}
= 
\begin{bmatrix}
  A - BK & BK \\
  0 & A - L_c CA
\end{bmatrix}
\begin{bmatrix}
  x[k] \\
  \tilde{x}[k]
\end{bmatrix}.
$$

As in the prediction estimator case, the closed-loop poles of our compensated system are the eigenvalues of

$$
\text{poles} = \text{eig} \left[ \begin{array}{cc}
  A - BK & BK \\
  0 & A - L_c CA
\end{array} \right]
$$

or

$$
\chi_{cl}(z) = \chi_{des}(z) \cdot \chi_{ob,des}(z)
= \text{det}(zI - A + BK) \text{det}(zI - A + L_c CA).
$$

Therefore

$$
u[k] = -K \hat{x}_c[k]
$$

also works!

What is $$D(z)$$ for the current estimator?

Consider a recurrence relation given earlier

$$
\hat{x}_c[k + 1] = (I - L_c C)(A - BK)\hat{x}_c[k] + L_c y[k + 1]
$$

$$
u[k] = -K \hat{x}_c[k].
$$

Taking the $$z$$-transform ($$\hat{x}_c[0] = 0$$)

$$
z \hat{X}_c(z) = (I - L_c C)(A - BK) \hat{X}_c(z) + zL_c Y(z)
$$

$$
U(z) = -K \hat{X}_c(z)
$$

so
\[
D(z) = \frac{U(z)}{Y(z)} = -K \left( zI - (I - L_c C)(A - BK) \right)^{-1} L_c z
\]

or

\[
D(z) = -K \left( zI - A + BK + L_c CA - L_c CBK \right)^{-1} L_c z.
\]

- Extra term in \((-1)\) and there is always a zero at \(z = 0\)!

- So we always end up with a compensator zero at the origin. The current compensator poles satisfy

\[
\det \left( zI - A + BK + L_c CA - L_c CBK \right) = 0.
\]

- Block diagram:

\[
\begin{array}{c}
\downarrow
\end{array} \quad \begin{array}{c}
L_c \quad \begin{array}{c}
\downarrow
\end{array} \quad zI \quad \begin{array}{c}
\oplus
\end{array} \quad z^{-1} \quad \begin{array}{c}
\downarrow
\end{array} \quad -K \quad \begin{array}{c}
\downarrow
\end{array} \quad u[k]
\end{array}
\]

\[
\begin{array}{c}
(I - L_c C)(A - BK)
\end{array}
\]

- We cannot implement the \(zI\) block. To see a more useful block diagram, let’s write the compensator recurrence relations in standard form, \(i.e.,\) let’s find

\[
x_{cc}[k + 1] = Ax_{cc}[k] + By[k]
\]

\[
u[k] = Cx_{cc}[k] + Dy[k].
\]

- Start with the control equation

\[
u[k] = -K \hat{x}_c[k]
\]

\[
= -K \left( \hat{x}_p[k] + L_c \left( y[k] - C \hat{x}_p[k] \right) \right)
\]

\[
= -K \left( I - L_c C \right) \hat{x}_p[k] - KL_c y[k].
\]
Recursion for $\hat{x}_p[k]$?

\[
\hat{x}_p[k + 1] = A\hat{x}_p[k] + Bu[k] + AL_c \left( y[k] - C \hat{x}_p[k] \right)
= (A - AL_c C) \hat{x}_p[k] - BK \hat{x}_c[k] + AL_c y[k]
= (A - AL_c C - BK + BKL_c C) \hat{x}_p[k] + (AL_c - BKL_c) y[k].
\]

Therefore

\[
\hat{x}_p[k + 1] = \tilde{A}\hat{x}_p[k] + \tilde{B} y[k]
\]
\[
u[k] = \tilde{C} \hat{x}_p[k] + \tilde{D} y[k]
\]

where
\[
\tilde{A} = (A - BK)(I - L_c C) \quad \tilde{C} = -K(I - L_c C)
\]
\[
\tilde{B} = (A - BK) L_c \quad \tilde{D} = -KL_c
\]

Our modified transfer function for $D(z)$ is

\[
D(z) = \tilde{D} + \tilde{C}(zI - \tilde{A})^{-1}\tilde{B}
\]

or

\[
D(z) = -KL_c - K(I - L_c C) \left( zI - (A - BK)(I - L_c C) \right)^{-1} (A - BK) L_c.
\]
which you can verify is equivalent to the previous expression.

New block diagram
Because of delay, everything inside the dotted box can be computed before we sample $y[k]$.

Note the feedthrough term $-KL_c$. So, the compensator responds quickly to plant variations. That is,

$$u[k] = f(y[k], y[k - 1], \ldots).$$

**IMPLEMENTATION METHOD 1:** (not good)

\[ \begin{align*}
\hat{x}_{tp} &= \hat{x}_{tp}\text{new} \\
A_2D(y) \\
\hat{x}_{tc} &= \hat{x}_{tp} + L_c*(y - C*\hat{x}_{tp}) \\
u &= -K*\hat{x}_{tc} \\
D_2A(u) \\
\hat{x}_{tp}\text{new} &= A*\hat{x}_{tc} + B*u
\end{align*} \]

**IMPLEMENTATION METHOD 2:** (good)

\[ \begin{align*}
\hat{x}_{tp} &= \hat{x}_{tp}\text{new} \\
upartial &= -K*(I-L_c*C)*\hat{x}_{tp} \\
A_2D(y) \\
u &= upartial - K*L_c*y \\
D_2A(u) \\
\hat{x}_{tp}\text{new} &= A*\hat{x}_{tp} + B*u + A*L_c*(y - C*\hat{x}_{tp})
\end{align*} \]

**EXAMPLE:** $G(s) = \frac{1}{s^2}$; $T = 1.4$ seconds.

- System description
  \[ A = \begin{bmatrix} 1 & 1.4 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1.4 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \]

- Pick closed-loop poles as we did for prediction estimator
  \[ \chi_{des}(z) = z^2 - 1.6z + 0.7 \quad \text{and} \quad \chi_{ob.des}(z) = z^2 - 1.28z + 0.45. \]
- Control design $K = \begin{bmatrix} 0.05 & 0.25 \end{bmatrix}$.

- Estimator design

$$L_c = A^{-1}L_p = A^{-1} \begin{bmatrix} 0.72 \\ 0.12 \end{bmatrix} = \begin{bmatrix} 0.55 \\ 0.12 \end{bmatrix}.$$ 

- Our compensator is described by

$$\hat{x}_p[k + 1] = \begin{bmatrix} 0.29 & 1.16 \\ -0.11 & 0.65 \end{bmatrix} \hat{x}_p[k] + \begin{bmatrix} 0.66 \\ 0.04 \end{bmatrix} y[k]$$

$$u[k] = \begin{bmatrix} 0.0067 & -0.25 \end{bmatrix} \hat{x}_p[k] - 0.06y[k]$$

or

$$D(z) = -0.06 \frac{z(z - 0.85)}{z - 0.47 \pm 0.31j}$$

$$= -0.06 \frac{z(z - 0.85)}{z^2 - 0.94z + 0.316}.$$ 

- The root locus

- Compare to prediction estimator.

**EXAMPLE:** In MATLAB,
\[ Q = [A -B*K; Lc*C*A, A-B*K-Lc*C*A]; \]

- get resp. of \( u, x, \hat{x} \) to init. condition

\[
[u, X] = \text{dinitial}(Q, \text{zeros}([4, 1]), [0 0 -K], 0, x0);
\]

- get the estimate error.

\[ \hat{x} = X(:, 1:2) - X(:, 3:4); \]

- Compare to prior result:
7.8: Discrete-time reduced-order estimator

- Why construct the entire state vector when you are directly measuring a state? If there is little noise in your sensor, you get a great estimate by just letting

\[ \hat{x}_1 = y \quad (C = [1 \ 0 \ \ldots \ 0]). \]

- If there is noise in the measurement

\[ y[k] = Cx[k] + v, \quad v = \text{noise}, \]

then the estimate \( \hat{y}_p \) or \( \hat{y}_c \) can be a smoothed version of \( y \! \).

- Consider partitioning the plant state into

\[ x_a : \text{measured state} \]
\[ x_b : \text{to be estimated}. \]

So

\[ y = Cx = x_a. \]

- (Note: This may require a transformation).

- Our partitioned system

\[
\begin{bmatrix}
    x_a[k+1] \\
    x_b[k+1]
\end{bmatrix} =
\begin{bmatrix}
    A_{aa} & A_{ab} \\
    A_{ba} & A_{bb}
\end{bmatrix}
\begin{bmatrix}
    x_a[k] \\
    x_b[k]
\end{bmatrix} +
\begin{bmatrix}
    B_a[k] \\
    B_b[k]
\end{bmatrix} u[k]
\]

\[ y[k] = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} x_a[k] \\ x_b[k] \end{bmatrix} \]

- In order to design an estimator for \( x_b \), we need to create a suitable state-space model of the dynamics of \( x_b \) such as
$x_b[k + 1] = A_{xb}x_b[k] + B_{xb}m_1[k]$

$m_2[k] = C_{xb}x_b[k] + D_{xb}m_1[k]$

where $m_1[k]$ and $m_2[k]$ are some measurable inputs. Note that $m_1[k]$ and $m_2[k]$ will be combinations of $y[k] = x_a[k]$ and $u[k]$.

- Once we have this state-space form, we can create a standard prediction or current estimator.

- We start the derivation by finding an output equation for $x_b$. Consider the dynamics of the measured state:

  $x_a[k + 1] = A_{aa}x_a[k] + A_{ab}x_b[k] + B_a u[k]$

  where $x_b[k]$ is the only unknown. Let

  $m_2[k] = x_a[k + 1] - A_{aa}x_a[k] - B_a u[k]$.

- Then

  $m_2[k] = A_{ab}x_b[k]$,

  where $m_2[k]$ is known/measurable and thus “$C_{xb}$” is equal to $A_{ab}$.

- This is our reduced-order estimator output relation.

- We now look for a state equation for $x_b$. Consider the dynamics of the estimated state:

  $x_b[k + 1] = A_{ba}x_a[k] + A_{bb}x_b[k] + B_b u[k]$.

  Let

  $B_{xb}m_1[k] = A_{ba}x_a[k] + B_b u[k]$  

  so that the reduced-order recurrence relation is

  $x_b[k + 1] = A_{bb}x_b[k] + B_{xb}m_1[k]$. 
This might be accomplished via (single-input system)

\[
B_{xbm_1}[k] = \begin{bmatrix}
A_{ba} & B_b
\end{bmatrix}
\begin{bmatrix}
x_a[k] \\
u[k]
\end{bmatrix}
\]

although the details of how this is done do not matter in the end.

So, for the purpose of designing our estimator, the state-space equations are:

\[
x_b[k + 1] = A_{bb}x_b[k] + B_{xbm_1}[k]
\]

\[
m_2[k] = A_{ab}x_b[k].
\]

In terms of our standard estimator design procedures,

\[
A \leftarrow A_{bb}; \quad Bu[k] \leftarrow B_{xbm_1}[k]; \quad C \leftarrow A_{ab}; \quad y \leftarrow m_2[k].
\]

For example, we can design a prediction reduced-order estimator. Once we know \(L_r\), we have the estimator equation

\[
\hat{x}_b[k + 1] = A_{bb}\hat{x}_b[k] + B_{xbm_1}[k] + L_r \left( m_2[k] - A_{ab}\hat{x}_b[k] \right).
\]

In terms of our known (measured) quantities,

\[
\hat{x}_b[k + 1] = A_{bb}\hat{x}_b[k] + A_{ba}x_a[k] + B_bu[k] +
L_r \left( x_a[k + 1] - A_{aa}x_a[k] - B_au[k] - A_{ab}\hat{x}_b[k] \right).
\]

Note that when we look at the compensator, we will be able to make the \(x_a[k + 1]\) disappear.

In order to design \(L_r\) we must find the equation that the error dynamics satisfy.

\[
\tilde{x}_b[k + 1] = x_b[k + 1] - \hat{x}_b[k + 1]
\]
\[ E_{bb}x_b[k] + B_{xb}m_1[k] - \\
(A_{bb}\hat{x}_b[k] + B_{xb}m_1[k] + L_r(A_{ab}x_b[k] - A_{ab}\hat{x}_b[k])) \\
= (A_{bb} - L_rA_{ab})\hat{x}_b[k]. \]

- So we pick estimate error dynamics related to roots of

\[ \chi_{r,des}(z) = \det(zI - A_{bb} + L_rA_{ab}) = 0. \]

- You might guess that arbitrary reduced-order estimator poles can be selected if \( \{A_{ab}, A_{bb}\} \) forms an observable pair.

**Design of \( L_r \)**

- Relate coefficients of

\[ \det(zI - A_{bb} + L_rA_{ab}) = \chi_{r,des}(z). \]

- Ackermann’s formula with

\[
L_r = \chi_{r,des}(A_{bb}) \begin{bmatrix}
A_{ab} \\
A_{ab}A_{bb} \\
\vdots \\
A_{ab}A_{bb}^{n-1}
\end{bmatrix}^{-1} \begin{bmatrix}
0 \\
\vdots \\
0 \\
1
\end{bmatrix}.
\]

- In MATLAB,

\[
\begin{align*}
Lr &= \text{acker}(Abb', Aab', poles)' \\
Lr &= \text{place}(Abb', Aab', poles)';
\end{align*}
\]
7.9: Discrete-time reduced-order prediction compensator

- We now determine (1) the closed-loop system dynamics using the reduced-order compensator, and (2) the reduced-order discrete-time compensator $D(z)$.

- We first combine the control law, the plant dynamics, and the estimator dynamics to find the closed-loop system dynamics.

- Control law:

$$u[k] = -K_a x_a[k] - K_b \hat{x}_b[k]$$

$$= -\begin{bmatrix} K_a & K_b \end{bmatrix} \hat{x}[k].$$

- Plant:

$$x[k + 1] = Ax[k] + Bu[k]$$

$$y[k] = C x[k] = x_a[k].$$

- Estimator:

$$\hat{x}_b[k + 1] = A_{bb} \hat{x}_b[k] + A_{ba} x_a[k] + B_b u[k] +$$

$$L_r (x_a[k + 1] - A_{aa} x_a[k] - B_a u[k] - A_{ab} \hat{x}_b[k])$$

using

$$u[k] = -K_a C x[k] - K_b \hat{x}_b[k]$$

$$L_r x_a[k + 1] = L_r C x[k + 1]$$

$$= L_r C (Ax[k] + Bu[k])$$

$$= L_r C A x[k] - L_r B_a K_a C x[k] - L_r B_a K_b \hat{x}_b[k].$$
\[
\begin{bmatrix}
  x[k + 1] \\
  \hat{x}_b[k + 1]
\end{bmatrix} = 
\begin{bmatrix}
  A - BK_a C & -BK_b \\
  L_r CA + A_{ba} C - B_b K_a C - L_r A_{aa} C & A_{bb} - B_b K_b - L_r A_{ab}
\end{bmatrix}
\begin{bmatrix}
  x[k] \\
  \hat{x}_b[k]
\end{bmatrix}
\]

- We can also investigate the closed-loop dynamics in terms of the estimator error state

\[
\begin{align*}
  x[k + 1] &= A x[k] - BK_a x_a[k] - BK_b (x_b[k] - \hat{x}_b[k]) \\
  \hat{x}_b[k + 1] &= (A_{bb} - L_r A_{ab}) \hat{x}_b[k],
\end{align*}
\]

or

\[
\begin{bmatrix}
  x[k + 1] \\
  \hat{x}_b[k + 1]
\end{bmatrix} = 
\begin{bmatrix}
  A - BK & BK_b \\
  0 & A_{bb} - L_r A_{ab}
\end{bmatrix}
\begin{bmatrix}
  x[k] \\
  \hat{x}_b[k]
\end{bmatrix}
\]

- So, we see that the separation principle holds when using the reduced-order estimator.

- We now proceed to find the compensator. We combine the estimator state equation and controller output equation.

\[
\begin{align*}
  \hat{x}_b[k + 1] &= (A_{bb} - B_b K_b + L_r B_a K_b - L_r A_{ab}) \hat{x}_b[k] \\
  &\quad + (A_{ba} + L_r B_a K_a - L_r A_{aa} - B_b K_a) y[k] \\
  &\quad + L_r y[k + 1] \\
  \hat{x}_b[k + 1] &= \bar{A} \hat{x}_b[k] + \bar{B} y[k] + L_r y[k + 1],
\end{align*}
\]

and

\[
u[k] = -K_b \hat{x}_b[k] - K_a y[k],
\]

so taking the z-transform
\[ D(z) = \frac{U(z)}{Y(z)} = -K_a - K_b (zI - \bar{A})^{-1} (\bar{B} + zL_r). \]

**Example:** \( G(s) = \frac{1}{s^2}; \) \( T = 1.4 \) seconds.

- **System description**
  \[ A = \begin{bmatrix} 1 & 1.4 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1.4 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \]

- **Measure** \( y[k] = x_1[k] \) directly; estimate \( v[k] = x_2[k] \).

- **Pick control poles the same as before:** \( \chi_{\text{des}}(z) = z^2 - 1.6z + 0.7 \).

- **Choose “dead-beat” estimation of** \( v[k] \): \( \chi_{r,\text{des}}(z) = z \).

- **Control design:** \( K = \begin{bmatrix} 0.05 & 0.25 \end{bmatrix} \).

- **Reduced-order estimator**
  \[ \chi_r(z) = \text{det}(zI - A_{bb} + L_r A_{ab}) = z - 1 + L_r T \]

  where
  \[ A = \begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix} = \begin{bmatrix} 1 & 1.4 \\ 0 & 1 \end{bmatrix} \]

  so \( L_r = \frac{1}{T} = 0.714 \) implies
  \[ \chi_r(z) = \text{det}(z - 1 + 0.714 \cdot 1.4) = z. \]

- **Reduced-order estimator:**
  \( A \) as above, \( B_a = \frac{T^2}{2} = 0.98; \quad B_b = T = 1.4. \)

  implies
  \[ \hat{v}[k + 1] = \hat{v}[k] + Tu[k] + \frac{1}{T} \left( y[k + 1] - y[k] - \frac{T^2}{2} u[k] - T \hat{v}[k] \right) \]

  \[ = \frac{y[k + 1] - y[k]}{1.4} + 0.7u[k]. \]
Control:

\[ u[k] = -\frac{1}{20} y[k] - \frac{1}{4} \hat{v}[k], \]

so

\[ \hat{v}[k + 1] = -\frac{T}{8} \hat{v}[k] + \frac{1}{T} y[k + 1] - \left( \frac{1}{T} + \frac{T}{40} \right) y[k] \]

\[ u[k] = -\frac{1}{4} \hat{v}[k] - \frac{1}{20} y[k]. \]

Taking the transfer function:

\[ D(z) = \frac{U(z)}{Y(z)} = -\frac{1}{20} - \frac{1}{4} \frac{1}{z + \frac{T}{8}}, \]

or

\[ D(z) = -\frac{1}{20} \left( 1 + \frac{5}{T} \right) \frac{z - \frac{5}{1 + \frac{5}{T}}}{z + \frac{T}{8}}. \]

In our case, \( T = 1.4, \)

\[ D(z) = -0.229 \frac{z - 0.781}{z + 0.175} \]
7.10: Continuous-time reduced-order estimator

- Here, we set $x_a$ to be the measured portion of the state and $x_b$ to be the unmeasured portion of the state. Then,

\[ x(t) = \begin{bmatrix} x_a(t) \\ x_b(t) \end{bmatrix}, \quad \text{and} \quad \hat{x}(t) = \begin{bmatrix} x_a(t) \\ \hat{x}_b(t) \end{bmatrix}. \]

- The state equations that the system must satisfy are

\[
\dot{x}(t) = \begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix}\begin{bmatrix} x_a(t) \\ x_b(t) \end{bmatrix} + \begin{bmatrix} B_a \\ B_b \end{bmatrix}u(t)
\]

\[
y(t) = \begin{bmatrix} I & 0 \end{bmatrix}\begin{bmatrix} x_a(t) \\ x_b(t) \end{bmatrix}.
\]

- We need to manipulate this form into a state-space system description of the unmeasurable $x_b(t)$ so that we can use standard tools to design an estimator $\hat{x}_b(t)$.

- First, we notice that the measured state $x_a(t)$ has dynamics

\[
\dot{x}_a(t) = A_{aa}x_a(t) + A_{ab}x_b(t) + B_a u(t)
\]

where $x_b(t)$ is the only unknown.

- Let

\[
m_2(t) = \dot{x}_a(t) - A_{aa}x_a(t) - B_a u(t),
\]

so that we can create an “output equation” of the $x_b$ dynamics

\[
m_2(t) = A_{ab}x_b(t)
\]

where $m_2(t)$ is the measured/computed output value and $x_b$ is the estimated value.
Now, we proceed to find the “state equation” of the $x_b$ dynamics. Note

$$\dot{x}_b(t) = A_{bb}x_b(t) + [A_{ba}x_a(t) + B_bu(t)],$$

where the items in the square brackets form a measured input signal.

We design an estimator state equation as

$$\hat{x}_b(t) = A_{bb}\hat{x}_b(t) + A_{ba}x_a(t) + B_bu(t) +$$

$$L_r(m_2(t) - A_{ab}\hat{x}_b(t))$$

$$= A_{bb}\hat{x}_b(t) + A_{ba}x_a(t) + B_bu(t) +$$

$$L_r(\dot{x}_a(t) - A_{aa}x_a(t) - B_a u(t) - A_{ab}\hat{x}_b(t)).$$

We cannot implement this directly due to the offending $\dot{x}_a(t)$ term in the measurement update.

To make this problem disappear, define $w(t) = \hat{x}_b(t) - L_r x_a(t)$

$$\dot{w}(t) = \dot{x}_b(t) - L_r \dot{x}_a(t)$$

$$= A_{bb}\hat{x}_b(t) + A_{ba}x_a(t) + B_bu(t) -$$

$$L_r A_{aa}x_a(t) - L_r B_a u(t) - L_r A_{ab}\hat{x}_b(t).$$

$$= [A_{bb} - L_r A_{ab}] [\hat{x}_b(t) - L_r x_a(t)] +$$

$$[A_{bb} - L_r A_{ab}] L_r x_a(t) +$$

$$A_{ba}x_a(t) - L_r A_{aa}x_a(t) + [B_b - L_r B_a] u(t)$$

$$= [A_{bb} - L_r A_{ab}] w(t) +$$

$$[(A_{bb} - L_r A_{ab})L_r + A_{ba} - L_r A_{aa}] x_a(t) +$$

$$[B_b - L_r B_a] u(t)$$
\[
\dot{w}(t) = \tilde{A}w(t) + \tilde{B}_1x_a(t) + \tilde{B}_2u(t)
\]
\[
\hat{x}_b(t) = w(t) + L_r x_a(t).
\]

Therefore, the final state-space equations describing the dynamics of the estimator are:

\[
\dot{w}(t) = [A_{bb} - L_rA_{ab}] w(t) + [(A_{bb} - L_rA_{ab})L_r + A_{ba} - L_rA_{aa}] x_a(t) +
\]
\[
[B_b - L_r B_a] u(t)
\]
\[
\hat{x}_b(t) = w(t) + L_r x_a(t).
\]

The poles of the estimator are

\[
\det(sI - (A_{bb} - L_rA_{ab})) = 0
\]

and we can design \( L_r \) using, for example

\[
L_r = \text{acker}(Abb', Aab', \text{poles})';
\]
7.11: Estimator pole placement

- As was the case for finding the control law, the design of an estimator (for single-output plants) simply consists of
  1. Selecting desired estimator error dynamics.
  2. Solving for the corresponding estimator gain.

- In other words, we find $L_p$, $L_c$, or $L_r$ by first selecting the roots of
  $$\chi_{ob,des}(z) \text{ or } \chi_{r,des}(z).$$

- So, what estimator poles do we choose?

- If possible, pick estimator poles that do not influence transient response of control-law poles.

- We know
  $$\chi_{cl} = \chi_{des} \cdot \chi_{ob,des}.$$

- Since $\chi_{des}$ was chosen to meet transient specifications, try to pick $\chi_{ob,des}$ such that estimator dynamics die out before control-law dynamics.

- With no disturbance, the only job requirement for the estimator is to correct for the uncertainty of $x[0]$!

- This will be (near) immediate if we pick poles well inside unit circle.

- Pick estimator poles as fast as possible?
In control law design we were concerned with

<table>
<thead>
<tr>
<th>Fast response</th>
<th>Slower response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large intermediate states</td>
<td>versus</td>
</tr>
<tr>
<td>Large control effort</td>
<td></td>
</tr>
</tbody>
</table>

In estimator design, large intermediate states and large feedback signals (control effort) do not carry the same penalty since they are just computer signals!

That is, $L_p y[k]$ is not limited by actuator hardware!

Question: Why not pick very fast estimator poles?

Answer: Sensor noise and uncertainty.

Control law design: Transient response versus actuator effort.

Estimator design: Sensor-noise rejection versus process-noise rejection.

Consider the design of $L_p$: The plant is

$$x[k + 1] = Ax[k] + Bu[k] + B_w w[k]$$
$$y[k] = C x[k] + v[k].$$

$w[k]$: Process noise, plant disturbances, plant uncertainties
$v[k]$: Sensor noise, biases.

Estimator: $\hat{x}_p[k + 1] = A\hat{x}_p[k] + Bu[k] + L_p (y[k] - C\hat{x}_p[k])$.

Only knowledge of $v, w$ through $y[k]$. 
Estimator error dynamics:

\[ \tilde{x}[k + 1] = Ax[k] + Bu[k] + B_w w[k] - A\hat{x}_p[k] - Bu[k] - L_p \left( Cx[k] + v[k] - C\hat{x}_p[k] \right) \]

\[ = (A - L_p C) \tilde{x}[k] + B_w w[k] - L_p v[k]. \]

If \( v[k] = 0 \) then

- "Large" \( L_p \) produces fast poles of \( A - L_p C \). Fast correction of \( w[k] \) effects.
- That is, we believe our sensor more than our model.
- We rely heavily on feedback to keep \( \tilde{x}[k] \) small.
- High bandwidth to quickly compensate for disturbances.

If \( w[k] = 0 \) then

- "Small" \( L_p \) produces slow poles of \( A - L_p C \). Less amplification of sensor noise.
- We believe our model more than our sensor.
- Rely on model to keep \( \tilde{x}[k] \) small... open-loop estimation!
- Low bandwidth for good noise rejection (smoothing).

Therefore, we pick the estimator poles, or design the estimator gain by examining the tradeoff:

- If \( A - L_p C \) fast:
  - Small transient response effects.
  - Fast correction of model, disturbances.
  - Low noise rejection.
- If \( A - L_p C \) slow:
- Significant transient response effect.
- Slow correction of modeling errors, disturbances.
- High noise rejection.

- In general
  1. Place poles two to six times faster than controller poles and in well-damped locations. This will limit the estimator influence on output response.
  2. If the sensor noise is too big, the estimator poles can be placed as much as two times slower than controller poles.

- Notes about (2):
  - Controller may need to be redesigned since the estimator will strongly influence transient behavior.
  - These slow estimator dynamics will not be excited by our reference command if our signal is properly included!

**MIMO observer design**

- The MIMO observer design problem is the dual of the MIMO controller design problem.
- Therefore, if \( \{A^T, C^T\} \) is “controllable,” the controller design procedures return \( L = K^T \).

**Where to from here?**

- We now know how to place closed-loop poles anywhere we want.
- Where do we want them? Is there an optimal set of locations?
- We next preview LQR design, which attempts to answer this question.