Observer design

Observability matrix:

\[
0 = \begin{bmatrix}
C \\
CA \\
CA^2 \\
\vdots \\
CA^{n-1}
\end{bmatrix}
\]

(A is nxn.)

If states we don't measure directly can be estimated with an "observer" (also called an "estimator"), \( \text{rank}(0) = n \).

in MATLAB: \( 0 = \text{obsv}(A, C) \)

Basic Idea:

Imagine we have a good idea of the actual plant dynamics. We can run a simulation of what we expect the plant to do, \( \hat{x} \), along with corresponding expected output, \( \hat{y} \).
Now, if \( y \) (actual) \( \neq \) \( \hat{y} \) (expect) do not agree, we can use feedback to correct our model's estimate of \( \hat{x} \):

\[
\begin{align*}
\hat{x} &= A\hat{x} + Bu + L(y - \hat{y}) \\
\hat{y} &= C\hat{x}
\end{align*}
\]

Now, model updates states as:

\[
\begin{align*}
\dot{\hat{x}} &= A\hat{x} + Bu + L(y - C\hat{x}) \\
&= (A - LC)\hat{x} + Bu + Ly
\end{align*}
\]

\[
\begin{align*}
e_x &= x - \hat{x} \\
\dot{e_x} &= \dot{x} - \dot{\hat{x}}
\end{align*}
\]

Show that \( \dot{e_x} = (A - LC)e_x \).
\[
\dot{\mathbf{x}} = (A - LC)(\mathbf{x} - \mathbf{e}_x) + Bu + Ly = \dot{\mathbf{x}} - \dot{\mathbf{e}}_x
\]
\[
\dot{\mathbf{e}}_x = (A - LC)e_x - (A - LC)x - Bu - Ly + \dot{\mathbf{x}}
\]

\[
\dot{\mathbf{e}}_x = (A - LC)e_x - Ax + LCx - Bu - Ly + Ax + Bu
\]

\[
\dot{\mathbf{e}}_x = (A - LC)e_x + LCx - Ly
\]

\[
\dot{\mathbf{e}}_x = (A - LC)e_x + L(Cx - y)
\]

\[
\dot{\mathbf{e}}_x = (A - LC)e_x
\]

\[
\text{Dynamics of the error.}
\]

→ Set poles of \([sI - (A - LC)]\) to set "speed" of observer.

→ Rule of thumb, set poles to be 2-6x faster than "dominant poles" of the plant.

Example, pendulum:

\[
\begin{align*}
\dddot{\theta} + \kappa \ddot{\theta} &= T \\
\ddot{\theta} + \omega_n^2 \theta &= T
\end{align*}
\]

\[
\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}
\]

\[
\mathbf{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}
\]

Find \( L = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \)
\[
\text{det}\left[ sI - (A - LC) \right] = \text{poles of estimator dynamics}
\]

\[
\begin{bmatrix}
  s & 0 \\
  0 & s
\end{bmatrix} - \begin{bmatrix}
  0 - l_1 & 1 - 0 \\
  -w_n^2 - l_2 & 0
\end{bmatrix}
\]

\[
= \begin{bmatrix}
  s + l_1 & -1 \\
  w_n^2 + l_2 & s
\end{bmatrix}
\]

\[
\text{det} \rightarrow s(s + l_1) + (w_n^2 + l_2)
\]

\[
S^2 + Sl_1 + w_n^2 + l_2 = 0
\]

Now, say we want: \( s_1 = -10w_n, \ s_2 = -10w_n \)

\[(s + 10w_n)(s + 10w_n) = 0\]

\[
S^2 + 20w_nS + 100w_n^2 = 0
\]

\[
\therefore \begin{bmatrix}
l_1 = 20w_n \\
l_2 = 99w_n^2
\end{bmatrix}
\]

\[
L = \begin{bmatrix}
l_1 \\
l_2
\end{bmatrix}
\]

or: \( L = \text{acker}(A', C', pe)' \)

where: \( pe = \begin{bmatrix}
-w_n^2 \\
-w_n^2
\end{bmatrix} \)
Kalman filters

Given a process model: \( \dot{x}_k = Ax_k + Bu_k + w \)
And a measurement model: \( z_k = Hx_k + v \)

Iterate to find a least squares estimate of the state, \( \hat{x} \).

Assumptions: \( w \) and \( v \) are both Gaussian.

Overview:

- Kalman filtering is an iterative loop
  with 5 basic steps.
- To visualize the updates, consider a simple case
  where \( x \) has one state and \( H=I \).

Algorithm:

1. **Predictor ("time update")**: Blindly simulates dynamics
   \( \hat{x}_k^- = Ax_{k-1} + Bu_{k-1} \)
2. **Corrector ("measurement update")**: Use sensor data to tweak prediction
   \( K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \)
   when \( H=I \), \( K_k = (P_k^-)^{-1} \)
3. \( \hat{x}_k = \hat{x}_k^- + K_k (z - H\hat{x}_k^-) \)
4. \( P_k^- = AP_k^- A^T + Q \)
5. \( P_k = P_k^- - K_k H P_k^- \)