Observer-based LQR:

Section 9.4 of textbook

We have seen that state-feedback solves the LQR problem in a nice algorithmically efficient way, whereas output feedback complicates things quite a bit and the propose algorithms for output feedback don't enjoy the guarantees that state-feedback based design benefit from. So the natural question is whether we can use a reconstructed state from the output instead.
There are actually a few variations of this setup that are of interest:

\[ y = Cx \leftarrow \text{output/observed available for control.} \]

\[ y =Cx + \text{noise} \leftarrow \text{sensor noise} \]

\[ y = \tilde{C}x + \text{noise}. \]

we use the nominal \( \tilde{C} \) & the actual \( C \) is \( \tilde{C} = C + \Delta C \).

\[ \uparrow \text{uncertain observation.} \]
The basic idea of observers & filters is that of designing a \((\text{dynamical})\) system such that

\[ X \quad \xrightarrow{\delta} \quad \text{observer or filter} \quad \xrightarrow{} \quad Y \]

an estimate/approximated \(X\)

The estimated \(X\) can then be used for feedback ... or just an estimate of a state that is not directly observed.

Okay ... let us see how this works out...
Suppose

\[
\begin{align*}
\dot{x} &= Ax + Bu \\
y &= Cx
\end{align*}
\]

In order to "observe" \( x \) from \( y \), we consider the system

\[
\hat{x} = \hat{A} \hat{x} + Bu + L(y - C\hat{x})
\]

Let us see how this works out:
We conclude that

\[ (A, C) \text{ is controllable.} \]

Comparing

\[ A + Lc \quad \text{and} \quad A + B_k \]

\[ A + Lc \]

is Hurwitz then

\[ (A + Lc)(x) = (A + Lc)e \]

\[ e = x \]

\[ \frac{dx}{dt} = A(x) + B_ku_{(t)} \]

\[ e = x - \frac{(A + B_k - (A + B_k) + L)(y - Cx)}{\mu} \]
or equivalently if

\((C, A)\) is observable/detectable.

\((C, A)\) is observable if the eigenvalues of \(A + LC\)
can arbitrarily be assigned, using \(L\).

\((C, A)\) is detectable if the eigenvalues of \(A + LC\)
can be placed in the LHP.

Thus, if \((A, C)\) \((C, A)\) is observable/detectable,
we can design \(L\) s.t. \(\hat{x} \to x\). Now what?
we can take the output \(y\), reconstruct \(\hat{x}\),
& then use \(\hat{x}\) instead of \(x\) in any state-feedback design.
Will this work?

\[ \dot{x} = A x + B u, \]
\[ y = C x, \]
\[ \dot{x} = A x + B u + L (y - C \hat{x}), \]
\[ u = -K \hat{x} \]

Suppose we want to regulate \( x \to 0 \). (LQR setting)

\[ e = (A + LC) e, \]
\[ \dot{x} = A \hat{x} + BK \hat{x} = A x - BK (x - e), \]
\[ e = x - \hat{x} = (A - BK)x + BK e. \]
The form of this matrix is very important... if we want to determine the eigenvalues of

\[
\begin{bmatrix}
A + BK & e^B K \\
0 & A + LC
\end{bmatrix}
\]

eigenvalues of \( A + BK \)

eigenvalues of \( A + LC \).

which means that if \( A + BK \) and \( A + LC \) are Hurwitz,

\[ x \rightarrow 0, \quad \hat{x} \rightarrow x \]

This is called separation principle in observer-based design: the observer & controller can be designed separately.
The overall structure looks like

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

We can also write as follows:

\[
\begin{align*}
\mathbf{u} &= -K \hat{\mathbf{x}} \\
\hat{\mathbf{x}} &= (\mathbf{A} - BK - LC) \hat{\mathbf{x}} + \mathbf{L} \mathbf{y}(t) \\
\hat{\mathbf{x}}(s) &= (s \mathbf{I} - \mathbf{A} + BK + LC)^{-1} \mathbf{L} \mathbf{y}(s) \\
\Rightarrow \hat{\mathbf{x}}(s) &= -K \left( s \mathbf{I} - \mathbf{A} + BK + LC \right)^{-1} \mathbf{L} \mathbf{y}(s) \\
\Rightarrow \mathbf{y}(s) &= \mathbf{u}(s) = -K \left( s \mathbf{I} - \mathbf{A} + BK + LC \right)^{-1} \mathbf{L} \mathbf{y}(s)
\end{align*}
\]
A "dynamic" controller:

\[ K(s) = -K(sI - (A - BK - LC))^{-1} \]

State-space realization:

\[
\begin{bmatrix}
A_k & B_k \\
C_k & D_k
\end{bmatrix}
\]
We now aim to develop a parallel theory for the situation where there is noise in the measurement available to the controller...

\[ \dot{x} = Ax + Bu + Gw \]
\[ y = Cx + v \]

Before we proceed we need to say a bit more about what we mean by noise: