Controller Design using State Feedback and Observer

Model Based Development of Embedded Systems

2014
Overview

- **State-Space Feedback**
  - Allows to control several state variables simultaneously
  - Works if the system is *controllable*
  - Popular method: LQ design
  - Integral control can be added by simple ad hoc trick

- **Observer**
  - Often, not all states of the system are observable
  - We can design an observer
  - If the system is observable
  - Observer can be designed with prespecified poles
    - Poles determine how fast the observer will converge to correct state estimate

- **State-Space Feedback with Observer**
  - Obtained by combining the above methods
State-Space Feedback

- Inverted Pendulum Example:
  - We want to control both angle and speed
  - Check whether the system is controllable
    \[
    \text{rank}(\text{ctrb}(A_{\text{pend}},B_{\text{pend}})) \quad \text{should be 4.}
    \]

- `gen_lqr.m`
  
  \[
  Q = \text{eye}(4) ; \quad % \text{Make an identity matrix} \\
  Q(2,2) = 10 ; \quad % \text{define penalties for, e.g., speed and angle} \\
  Q(3,3) = 500 ; \\
  R = 1 ; \\
  K = \text{lqr}(A_{\text{pend}},B_{\text{pend}},Q,R) \quad % \text{calculate feedback matrix}
  \]

- Look at model in `state_feedback.mdl`
Adding Integral control

- In cases, where steady_state error is an issue
- Or just, to get smoother behavior
- Same setup for inverted pendulum
  
  • gen_lqr_int.m

  ```matlab
  Aext = [A_pend, zeros(4,1) ; 1 0 0 0 0]; % add a component to the state
  Bext = [B_pend ; 0];
  Cext = [C_pend, zeros(2,1)];
  Dext = [D_pend];

  Qext = zeros(5,5);  % Make a zero matrix
  Qext(1,1) = 10;
  Qext(3,3) = 500;
  Qext(5,5) = 10;
  R = 1;
  Kext = lqr(Aext,Bext,Qext,R)
  ```
Observer

- In general not all states are observable
- E.g., assume only position and angular speed observable
- Same setup for inverted pendulum
  - Check whether the system is controllable
    \[
    \text{rank} \left( \text{obsv}(A\_pend, C\_pend) \right) \quad \text{should be 4.}
    \]
  - See how fast the system is
  - Done by finding poles of the closed system:
    \[
    \text{eig}(A\_pend - B\_pend*K)
    \]
  - Define poles of the observer (should be faster than closed system)
    \[
    P = [-10 -12 -14 -16];
    \]
  - Construct the observer gain
    \[
    L = \text{place}(A\_pend', C\_pend', P);\]
Expressing the Observer:

- Inputs \((u,y)\)
- \(\dot{x}_{\text{hat}} = (A - LC) x_{\text{hat}} + Bu + Ly\)
- Output \(= x_{\text{hat}}\)

In MATLAB, letting input be \([u;y]\)

\[
\begin{align*}
P &= [-50 \ -51 \ -52 \ -53]; \\
L &= \text{place}(A\_\text{pend}',C\_\text{alt}',P)'; \\
A\_\text{obs} &= (A\_\text{pend} - L*C\_\text{alt}); \\
B\_\text{obs} &= [B\_\text{pend}, L]; \\
C\_\text{obs} &= \text{eye}(4); \\
D\_\text{obs} &= \text{zeros}(4,3);
\end{align*}
\]
About LQ

Inputs:

- Penalty matrix $Q$ for the state variables. Typically $Q$ is a diagonal matrix, with each entry giving a penalty for each variable.
- Penalty matrix $R$ for the control input(s). Typically a diagonal matrix.
- The `lqr(A,B,Q,R)` command in MATLAB computes the feedback matrix $K$ so that the total penalty

$$\int_0^\infty (x^T Q x + u^T R u) \, dt$$

is minimized.
About Poles

- A system, whose dynamics is given by
  \[ \frac{dx}{dt} = Ax + Bu \]
  has in general a number of (complex) poles (number is dim of A), which say how fast the system converges or diverges.
- A pole \( p \) means that one component of explicit expression for dynamics evolves like
  \[ e^{pt} \]
- Thus, a pole with positive real-part means divergence.
- Poles with large negative real-part means quick convergence.
- When designing observer, it is important that observer converges significantly faster than the observed system.
  - Poles can be 5-10 times larger (should all be negative)