EECS 106B/206B

Discussion 6

GSI: Valmik Prabhu

Action Items

- Concurrent Enrollment lab access?
- Homework 2 due this weekend
- Lab Due in 6 days!!!!
- Robot booking policy

Clarifications

- Gravity Matrix
- Workspace plotting
- tag_pub.py

State Feedback Control

$$\Sigma \left\{ \begin{array}{rcl} \dot{x}(t) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) + Du(t) \end{array} \right.$$

$$u = -Kx$$

How do we get the state?

Dead Reckoning

We know the initial condition, the inputs, and the system model

$$\hat{x} = A\hat{x} + Bu$$

This works surprisingly well. If A is stable, what happens to the error $\hat{x} - x$?

More generally, dead reckoning uses the initial position and velocity (measurements or estimations) to calculate future position

Observer state feedback (Luenberger observer)

We know the systems output dynamics, so we can predict the output

$$\hat{y} = C\hat{x}$$

We can compare that to the real (measured) output and do state feedback

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

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

Note that this is essentially *identical* to the state feedback control problem

Stochastic System

We have the following system

$$\frac{dx}{dt} = Ax + Bu + Fv, \qquad E\{v(s)v^T(t)\} = R_v(t)\delta(t-s),$$

$$y = Cx + w, \qquad E\{w(s)w^T(t)\} = R_w(t)\delta(t-s)$$

v and w are random, gaussian variables with the following distributions

$$\mathrm{pdf}(v) = \frac{1}{\sqrt[n]{2\pi}\sqrt{\det R_v}} e^{-\frac{1}{2}v^T R_v^{-1} v}, \quad \mathrm{pdf}(w) = \frac{1}{\sqrt[n]{2\pi}\sqrt{\det R_w}} e^{-\frac{1}{2}w^T R_w^{-1} w}$$

Optimal Convergence: LQR => Kalman Filter

We define $P(t) = E\{(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^T\}$

And solve the Riccati equation:

$$\frac{dP}{dt} = AP + PA^T - PC^T R_w^{-1}(t) CP + FR_v(t)F^T, \quad P[0] = E\{x[0]x^T[0]\}.$$

Here $L(t) = P(t)C^T R_w^{-1}$

If the variances R_w and R_v are constant, this is equal to an LQR controller with

$$R = R_w \qquad Q = FR_vF^T$$

Let's switch to discrete time

Consider a discrete-time linear system with dynamics

$$x[k+1] = Ax[k] + Bu[k] + Fv[k], \qquad y[k] = Cx[k] + w[k],$$

where $v[k]$ and $w[k]$ are Gaussian white noise processes satisfying

$$E\{v[k]\} = 0, \qquad E\{w[k]\} = 0,$$

$$E\{v[k]v^T[j]\} = \begin{cases} 0 & k \neq j \\ R_v & k = j, \end{cases} \qquad E\{w[k]w^T[j]\} = \begin{cases} 0 & k \neq j \\ R_w & k = j, \end{cases}$$

$$E\{v[k]w^T[j]\} = 0.$$

Discrete time Kalman filter

$$L[k] = AP[k]C^T(R_w + CP[k]C^T)^{-1},$$

$$P[k+1] = (A - LC)P[k](A - LC)^T + FR_v F^T + LR_w L^T$$

$$P_0 = E\{x[0]x^T[0]\}.$$

This is an iterative function! Easy to code

How do we implement this?

First, we predict the state and its covariance using our system model

- Prediction

$$X_k^- = A_{k-1}X_{k-1} + B_kU_k$$

$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1}$$

If there are no sensor measurements at this time step, this is all we do

How do we implement this?

Then, we *update* our prediction based on the data

- Update

$$V_k = Y_k - H_k X_k^-$$

$$S_k = H_k P_k^- H_k^T + R_k$$

$$K_k = P_k^- H_k^T S_k^{-1}$$

$$X_k = X_k^- + K_k V_k$$

$$P_k = P_k^- - K_k S_k K_k^T$$

What is H?

- H is a sensor model (similar to C)
- *Can be different for each sensor*

What is Y?

- The actual measurement

What is R?

- The current covariance matrix for our particular sensor (usually just set this to constant)

Why is this better than the closed form solution?

You can use many different sensors and *fuse* them together

Your sensor update rates can be different

Computers are discrete anyways

The extended Kalman filter

- Real-world systems are all *nonlinear*.
- Use nonlinear dynamics for the prediction step
- Linearize about the prediction and find L for the update

$$\frac{d\hat{x}}{dt} = f(\hat{x}, u) + L(\hat{x})(y - h(\hat{x}, u)),$$

- This can get badly conditioned if the error is too large