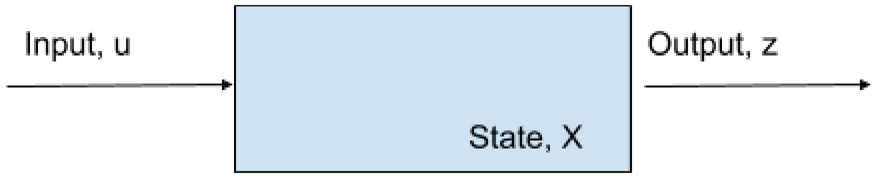
Lecture 5: State Estimators, Sensing

Video: Stickman, towards a Human Scale Acrobatic Robot. Disney Research.
(3 range sensors; state estimation. Robot itself cannot determine where it is in space without sensors in surrounding environment)

Lab 1: due 5pm Thu 1/24
Lab 2: due 5pm Thu 1/31
Preliminary Proposals due 5pm Thu 2/7

- *3 ideas*
- *Some background research done on each*

Recap of Class Material
A robot is a machine that we abstract as an input-output system.



The idea of a physical state: Given the size of the state we capture the translational/rotational state by setting a reference frame.
Examples:

- rigid body in free space (translational DOF=3, rotational DOF=3)
- robot cars (translational DOF=2, rotational DOF=1)

*translation -> minimal representation

Physical State:

Translation (2D/3D)
- Vector in reference frame

Rotation (1 DOF, 3 DOF)
- axis/angle
    - 3 DOF: 3 numbers to describe axis vector, 1 number for angle
- Quaternion
- DCM (direction cosine matrix)

*Rotation falls under non-minimal representation (unless 1 DOF). The representations are redundant

DCM
Pick 3 points on our object (for 3D object)

- Attach another coordinate plane and get new unit vectors
- n points, need n^2 to define our DCM
- How do we get back to 1DOF or 3DOF from n^2?
    - We have constraints:
        - for axis/angle, axis has no numbers (its always the same)
        - For quaternion, sum of all squares=1
        - For DCM, it is an orthonormal matrix, dot products of columns are 0

*minimal representation of rotational state -> euler's angles

(why did we invent concept of mathematical state?)
The concept of mathematical state is a history of inputs

$z(t) = f(u(0),\ u(1),\ ...,\ u(t)) \quad \rightarrow z(t) = h(x(t),\ u(t),\ t)$ Single set of values
Arbitrary number of inputs ^ ^create state to capture all the inputs

Dynamics defined as a of current state/input: $x' = f(x(t),\ u(t),\ t)$

Unfolding the system over time yields the following block diagram:

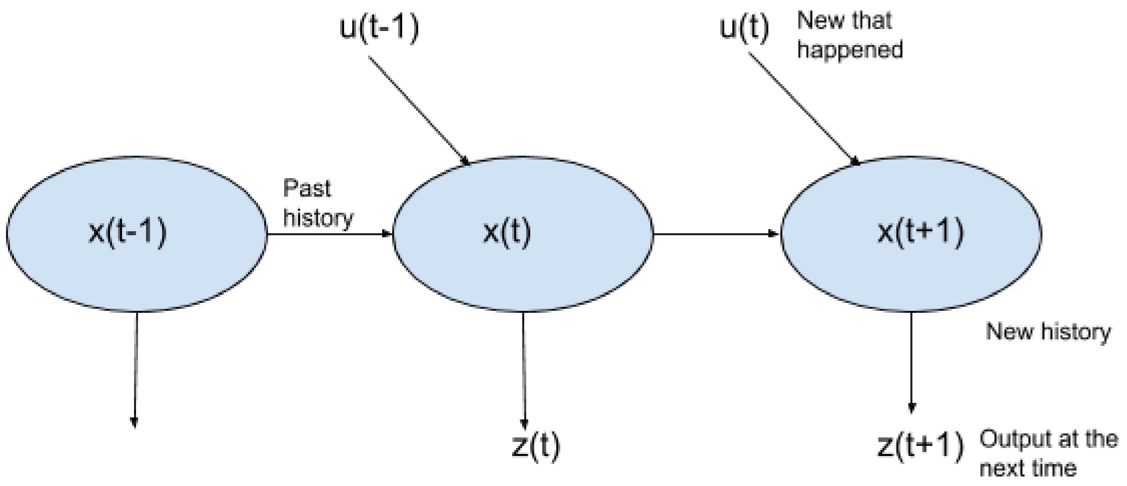


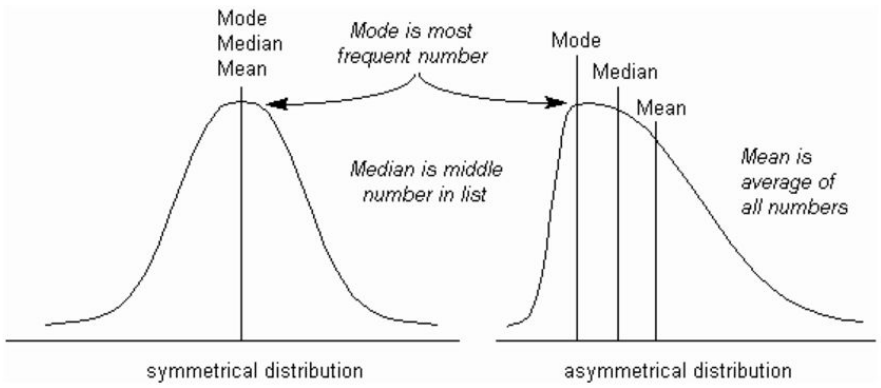
Fig. 2 Unfolding the system over time

The inside of the boxes is not known, that's why state estimation is needed.
State Estimation

Task: given $\{u, z\}$, compute the best guess for x
Defining the best guess includes describing it in terms of the Probability Function

- True value of state unknown, we describe the variability of the state with a probability distribution

Probability distribution of x


Mode
Median
Mean
Mode is most frequent number
Mode
Median
Mean
Median is middle number in list
Mean is average of all numbers
symmetrical distribution
asymmetrical distribution

$\bar{x}_{ML} = mode,\ maximum\ likelihood\ (peak)$
$\bar{x}_{MM} = median,\ min\ max$
$\bar{x}_{MV} = mean,\ minimum\ variance$

The focus is on the minimum variance or in other words how far from estimate the state may be.

<u>Minimum Variance State Estimate</u>

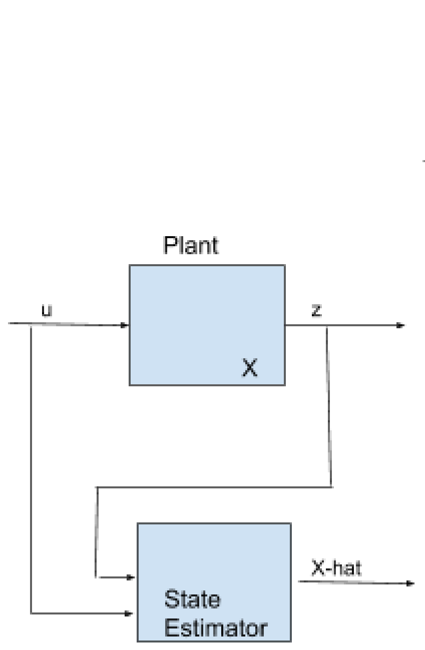
Let x be a true state, then $\hat{x} = \bar{x}_{MV} = arg_{\hat{x}}min\ E\left[(x-\hat{x})^{r}(x-\hat{x})\right]$
*We want to minimize the square of the error:
$"error"\ e = x - \hat{x}$; $\hat{x} = arg_{\hat{x}}min(|e|^{2})$ □ Quadratic

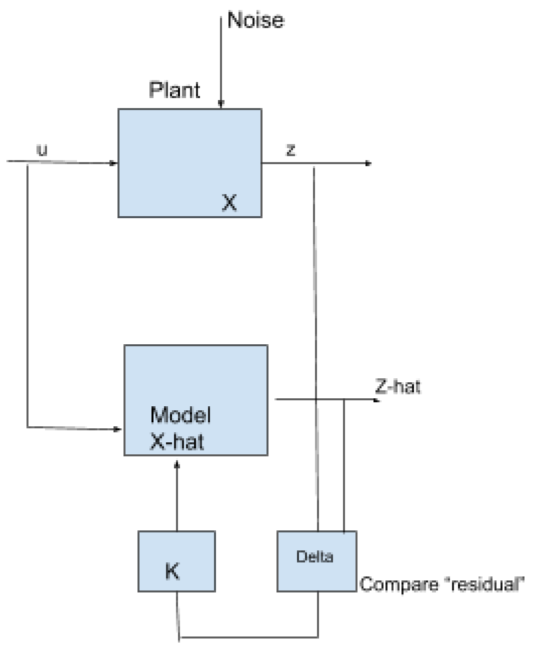
- We're minimizing some quadratic of the error, so we're building a quadratic state estimator

If linear system, <u>LQE</u>

<u>Building a Linear Quadratic State Estimator (LQE) = Kalman Filter</u>

How to minimize an error function using a linear system.





$\hat{x}[t+1] = A\hat{x}[t] + Bu[t] + d[t]$
$d[t]$: *process noise* ; $N(0,\ Q)$ (normal distribution, Q is variance)

Because errors will loop back and increase due to state being based on past states. Noise cannot be model, will affect state. Model the uncertainty.

$\hat{z}[t] = c\tilde{x}[t] + n[t]$
$n[t]$: *measurement noise* $\rightarrow N(0, R)$

*(n[t] only changes output, not state)
Good way to think of it is d[t] is caused by outside environment (i.e. actuator, wind, etc.)
n[t] is caused by measurement error (i.e. uncertainty in calibration, etc.)

$\tilde{z}[t] = z[t] - \hat{z}[t] = z[t] - c\,\hat{x}[t]$ ($\hat{z}[t]$ is the residual)
$\tilde{x}[t] = k\hat{z}[t] \rightarrow innovation$
$\hat{x}[t] \leftarrow \hat{x}[t] + \tilde{x}[t]$
*we get new u(input), plant and model update, then new z(output) updates model again

<u>A priori state update</u>: $\hat{x}^{-}(t+1) = A\hat{x}^{+}(t) + B\,u(t)$
$\hat{x}^{+}(t+1) = \hat{x}^{-}(t+1) + k\,\tilde{z}(t)$
k: Kalman Filter gain
$k = arg_{\ k}\ min\ E[e * e^{T}]$
$e = x(t) - \hat{x}^{+}(t) \rightarrow output\ information = best\ guess$
To minimize error take the derivative(with respect to k) , set it equal to zero, solve for k

*k will be a matrix, size n x m

- n is the size of x, m is the size of z

Error covariance

$P(t) = E[e(t) * e(t)^T]$

e is defined by e= x(t) - $\hat{x}^+(t)$

Kalman Filter(KF)

1) Time Update (also known as dynamics propagation, prediction, time propagation)

$$\hat{x}^-(t+1) = A\hat{x}^+(t) - B\,u(t)$$

$$p^-(t+1) = Ap^+(t) * A^T + a$$

* A Priori
* remember Q is covariance of d[t]

2) Measurement update

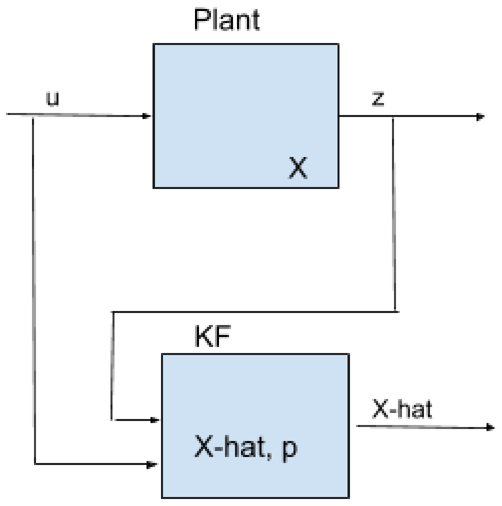
$$k = p^- C^T (cp^- C^T + R)^{-1}$$ (Kalman Filter gain, time indices dropped)

* the kalman filter gain above comes from taking the derivative of E[ee^T] with respect to t, setting to zero

$$\hat{x}^+ = \hat{x}^- + k(z - c\hat{x}^-)\,;\; k(z - c\hat{x}^-) = innovation\,;\; (z - c\hat{x}^-) = residual$$

*Posteriori

$$p^+ = (I - kc)p^-$$



The Kalman Filter is an optimal linear filter when the plant is linear because it minimizes the quadratic error with Gaussian Noise

Extended Kalman Filter

Used if plant is not linear. The system is time variant

* in extended, non-linear plant is linearized first

1) Time Update

$$\hat{x}^{-}(t+1) = A_t \hat{x}^{+}(t) - B_t u(t)$$

$$p^{-}(t+1) = A_t p^{+}(t) * A_t^{T} + a$$

2) Measurement update

$$k = p^{-} C^{T} (c_t p^{-} C_t^{T} + R)^{-1}$$

$$\hat{x}^{+} = \hat{x}^{-} + k(z - c_t \hat{x}^{-}) \text{ ; } k(z - c_t \hat{x}^{-}) = innovation \text{ ; } (z - c_t \hat{x}^{-}) = residual$$

$$p^{+} = (I - kc_t) p^{-}$$