Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

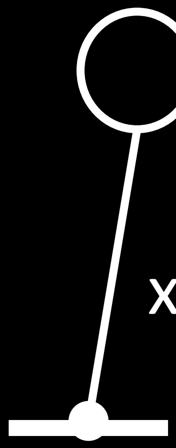
ECE 4160/5160
MAE 4910/5910

Fast Robots
Observability

Based on “Control Bootcamp”, Steve Brunton, UW
https://www.youtube.com/watch?v=Pi7l8mMjYVE

Linear Systems

- Linear systems review
- Eigenvectors and eigenvalues
- Stability
- Discrete time systems
- Linearizing non-linear systems
- Controllability
- LQR
- Observability



$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

$$\dot{x} = Ax + Bu$$

This should look familiar from..

- MATH 2940 Linear Algebra
- ECE3250 Signals and systems
- ECE5210 Theory of linear systems
- MAE3260 System Dynamics
- etc…

Linear Systems Control – "review of review"

- Linear system: $\dot{x} = Ax$
- Solution: $x(t) = e^{At}x(0)$
- Eigenvectors: $T = [\xi_1 \quad \xi_2 \quad \dots \quad \xi_n]$
- Eigenvalues: $D = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \dots & \\ 0 & & & \lambda_n \end{bmatrix}$

```
>>[T,D] = eig(A)
```

- Linear transform: $AT = TD$
- Solution: $e^{At} = Te^{Dt}T^{-1}$
- Mapping from x to z to x: $x(t) = Te^{Dt}T^{-1}x(0)$
- Stability in continuous time: $\lambda = a + ib$, stable iff a<0
  - Discrete time: $x(k+1) = \tilde{A}x(k), \tilde{A} = e^{A\Delta t}$
    - Stability in discrete time: $\tilde{\lambda}^n = R^n e^{in\theta}$, stable iff R<1

- Linearizing non-linear systems
  - Fixed points
  - Jacobian
- Controllability
  - $\dot{x} = (A - BK)x$
  - `>>rank(ctrb(A,B))`
- Reachability
- Controllability Gramian
- Pole placement
  - `>>K=place(A,B,p)`
- Optimal control (LQR)
  - `>>K=lqr(A,B,Q,R)`

Linear Quadratic Control

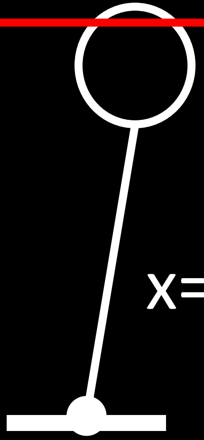
$$\dot{x} = Ax + Bu, x \epsilon \mathbb{R}^n$$
$$u = -Kx$$
$$\dot{x} = (A - BK)x$$

- >> K = place(A,B,eigs)
- Where are the best eigs??
  - Linear Quadratic Regulator (LQR)
    - >> K = lqr(A,B,Q,R)
    - Riccati equation
      - $\int_0^\infty (x^T Q x + u^T R u) dt$
    - $Q = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 10 & \\ 0 & & & 100 \end{bmatrix}, R = 0.01$

Based on "Control Bootcamp", Steve Brunton, UW
https://www.youtube.com/watch?v=Pi7l8mMjYVE

Linear Systems

- Linear systems review
- Eigenvectors and eigenvalues
- Stability
- Discrete time systems
- Linearizing non-linear systems
- Controllability
- LQR
- Observability



$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

$$\dot{x} = Ax + Bu$$

This should look familiar from..

- MATH 2940 Linear Algebra
- ECE3250 Signals and systems
- ECE5210 Theory of linear systems
- MAE3260 System Dynamics
- etc…

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

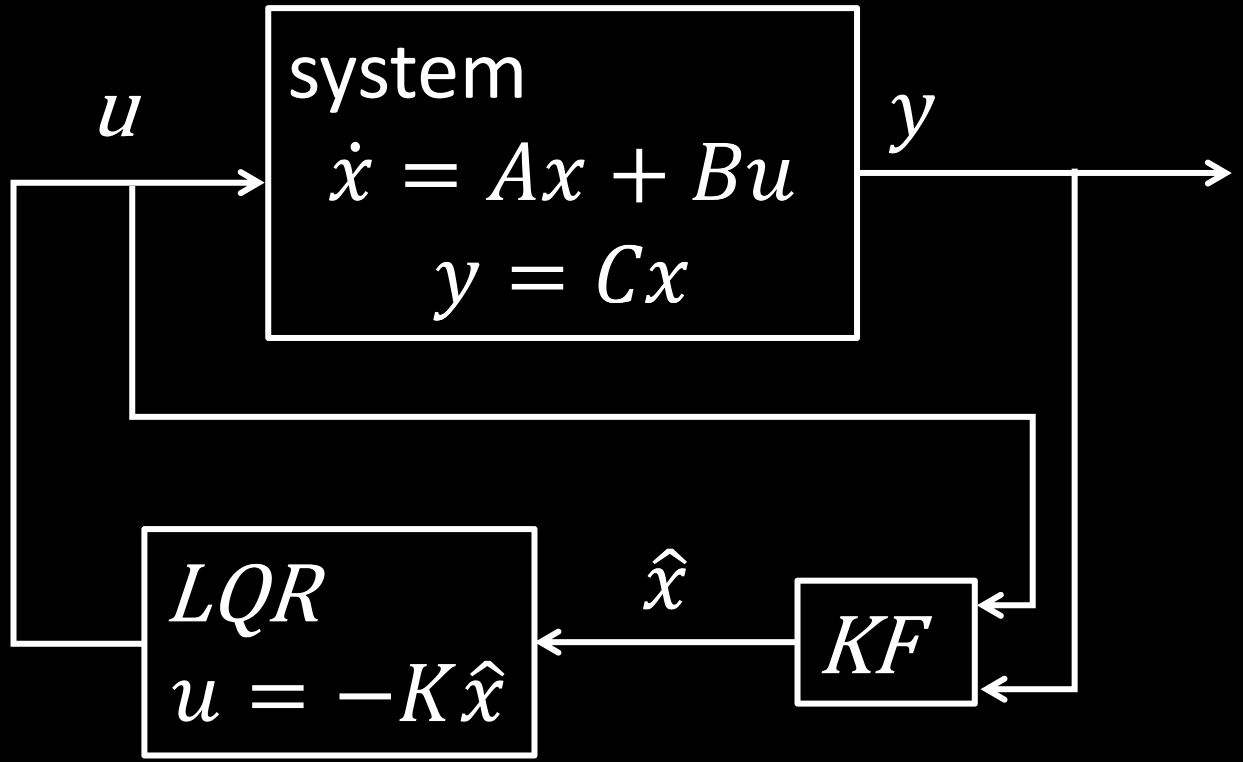
ECE 4160/5160
MAE 4910/5910

Fast Robots
Observability

Observability

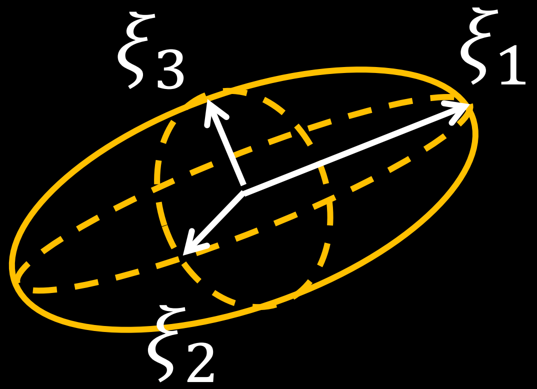
$$\dot{x} = Ax + Bu, x \epsilon \mathbb{R}^n$$
$$u = -Kx$$
$$\dot{x} = (A - BK)x$$

- Controllability
  - Can we steer the system anywhere given some control input u?
- Observability
  - Can we estimate any state x, from a time series of measurements y(t)?



Observability

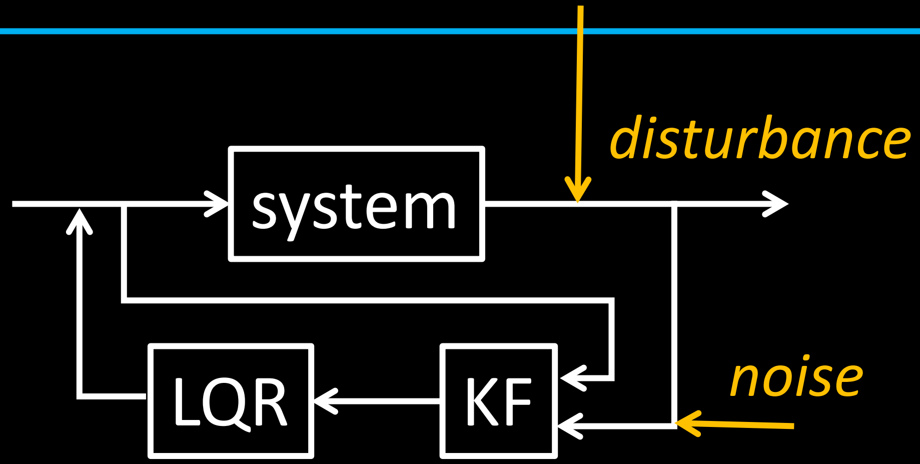
- $\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \dots \\ CA^{n-1} \end{bmatrix}$



1. Observable iff rank($\mathcal{O}$) = n
   - `>>rank(obsv(A,C))`
2. Iff a system Is observable, we can estimate x from y
   - Observability Gramian
     - `>>[U,Σ, V]=svd(`$\mathcal{O}$`)`

$$\dot{x} = Ax + Bu + d \qquad x \epsilon \mathbb{R}^n$$
$$y = Cx + n \qquad u \epsilon \mathbb{R}^q$$
$$y \epsilon \mathbb{R}^p$$

- Controllability
- $\mathbb{C} = [B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B]$
- >>ctrb(A,B)
- Reachability



ECE 4160/5160
MAE 4910/5910

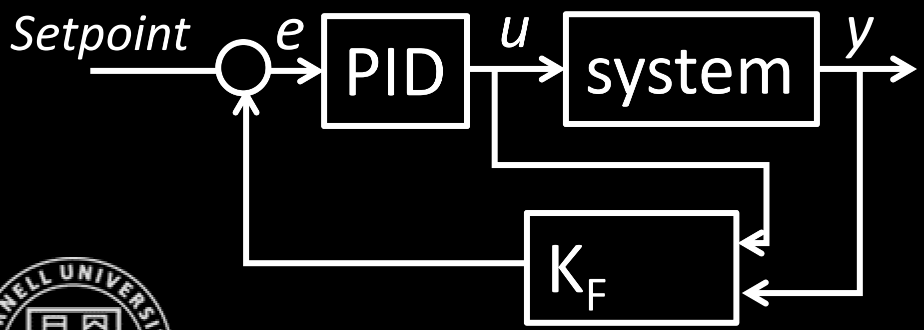
Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Kalman Filter

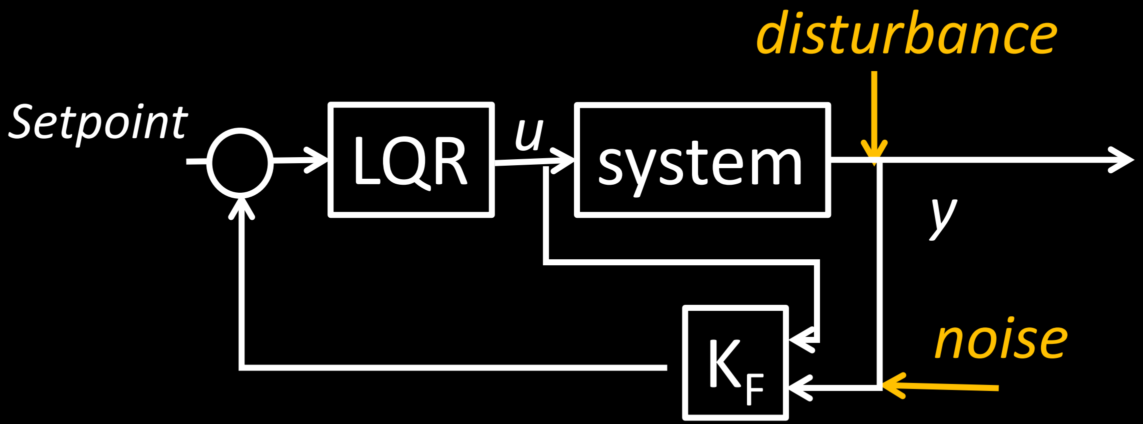
Why sensor fusion?

- Not full state feedback
- Bad sensors
- Imperfect model
- Slow feedback

KF with PID



What you typically apply KF on

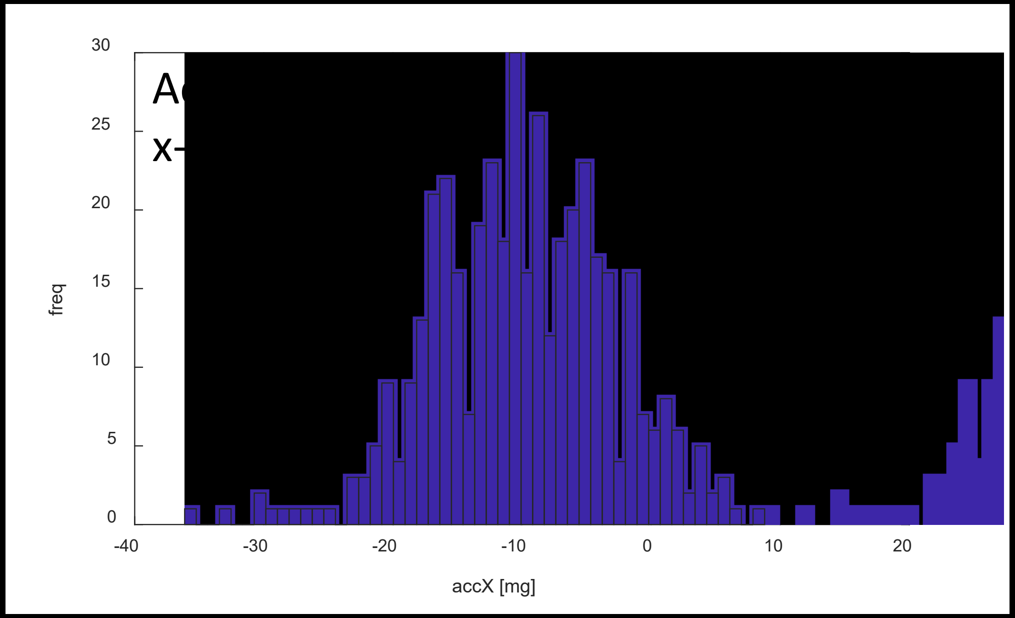


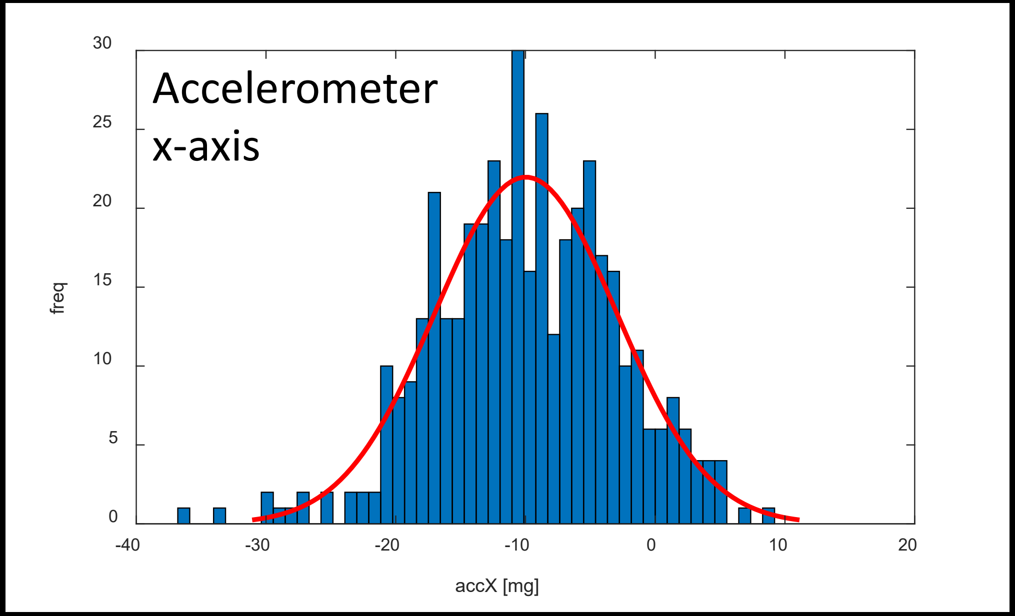
Probabilistic Robotics

Sources of uncertainty

- Measurements are uncertain
- Actions are uncertain
- Models are uncertain
- States are uncertain

- Gaussian distributions
    - $[\mu \mp \sigma]$
    - Symmetric
    - Unimodal
    - Sum to “unity”

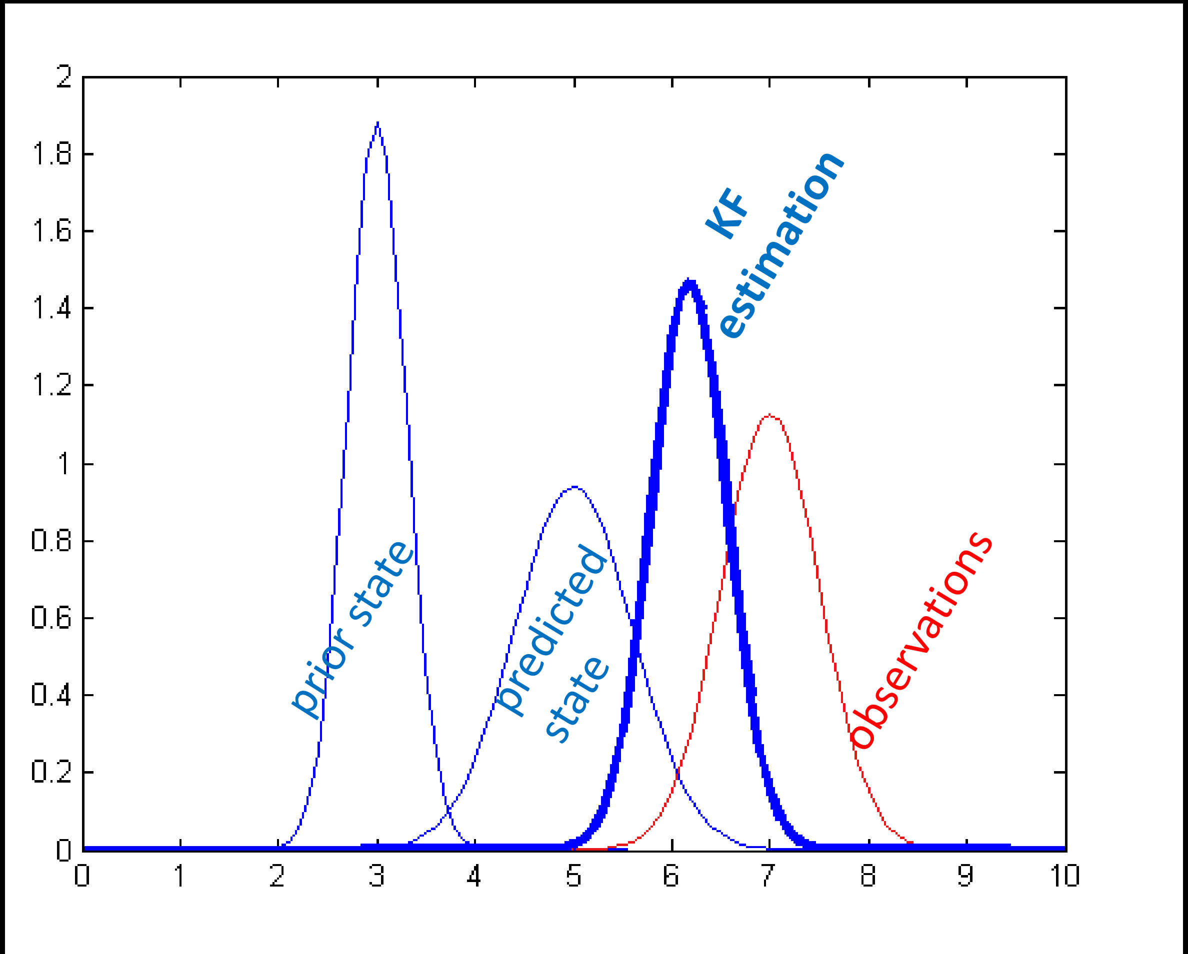




Kalman Filter

Incorporate uncertainty to get better estimates based on both inputs and observations

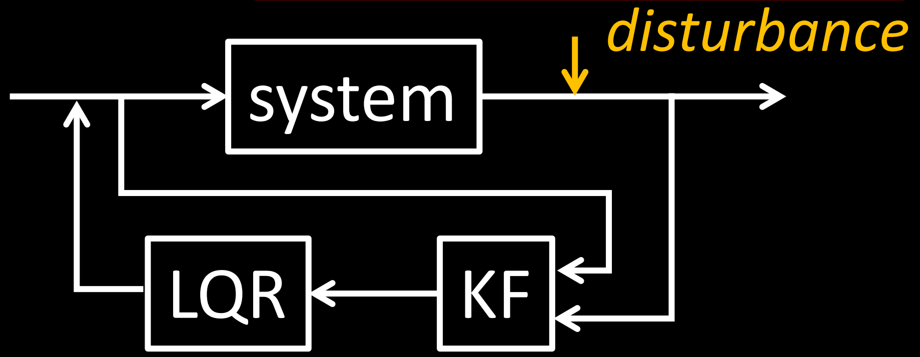
- Assume that posterior and prior belief are Gaussian variables

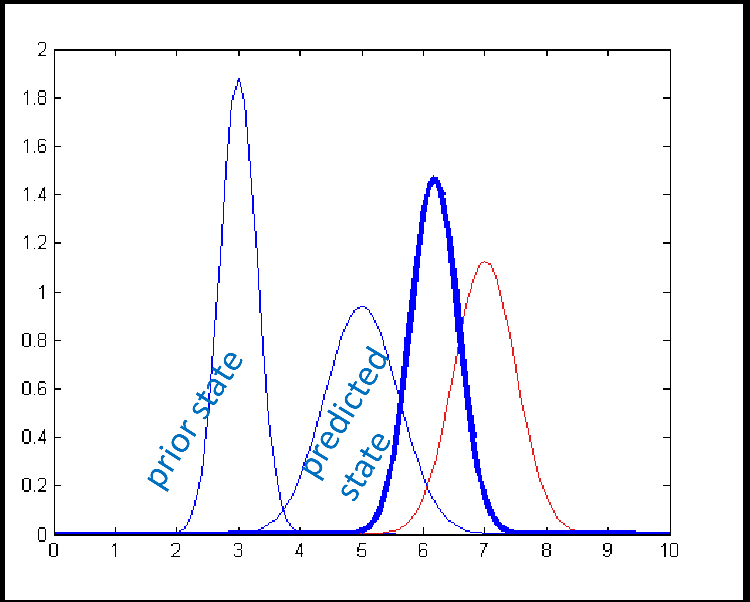


Kalman Filter

- Assume that posterior and prior belief are Gaussian variables
  - Prediction step
    - $x(t) = A\,x(t-1) + Bu(t) + n$, where…
      - $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
      - $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
  - Update step

State estimate: $\mu(t)$
State uncertainty: $\Sigma(t)$
Process noise: Σ_u

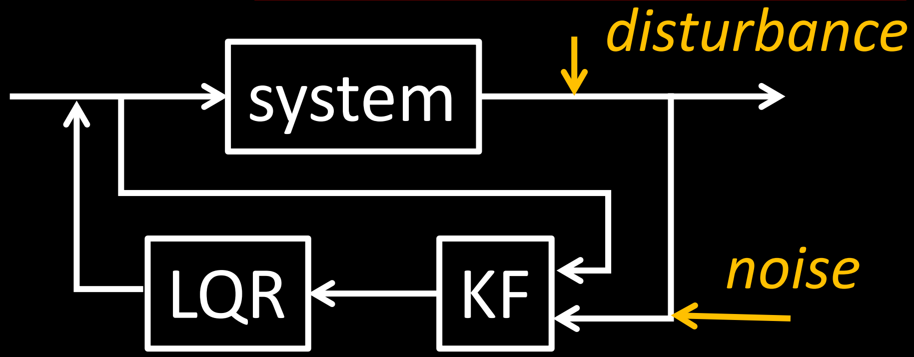


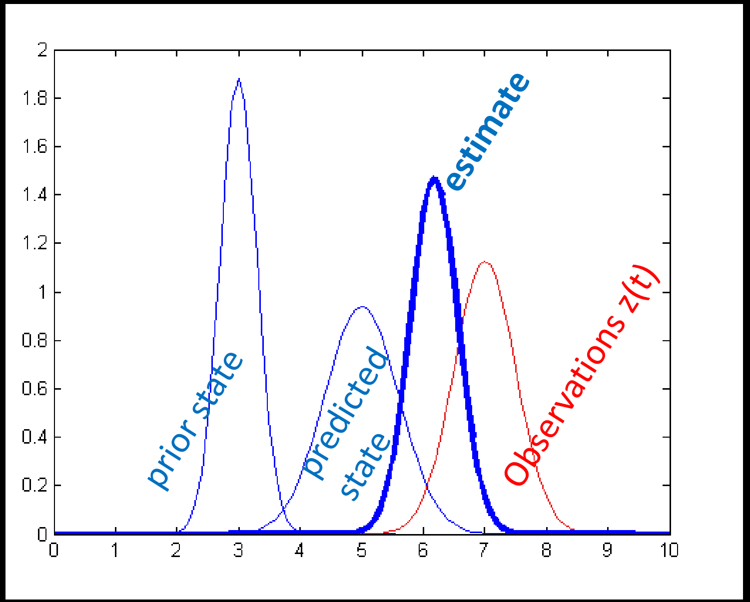


Kalman Filter

- Assume that posterior and prior belief are Gaussian variables
  - Prediction step
    - $x(t) = A\,x(t-1) + Bu(t) + n$, where…
      - $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
      - $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
  - Update step
    - $K_{KF} = \Sigma_p(t)\,C^T\,(\,C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
    - $\mu(t) = \mu_p(t) + K_{KF}\,(\,z(t) - C\,\mu_p(t)\,)$
    - $\Sigma(t) = (\,\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$

State estimate: $\mu(t)$
State uncertainty: $\Sigma(t)$
Process noise: Σ_u
Kalman filter gain: K_{KF}
Measurement noise: Σ_z





Kalman Filter

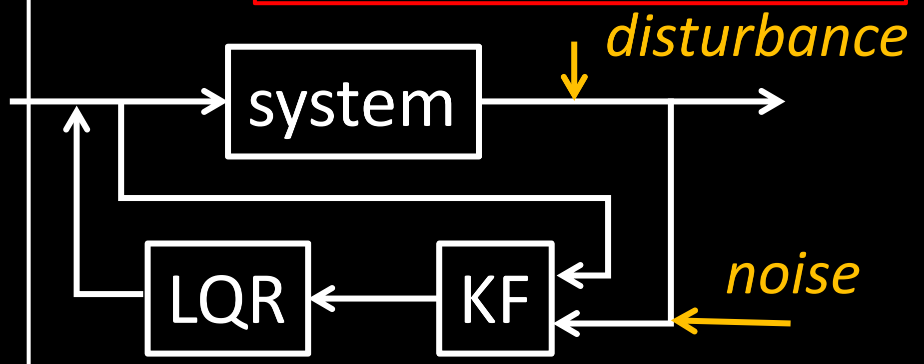
Function ($\mu(t-1)$, $\Sigma(t-1)$, $u(t)$, $z(t)$)

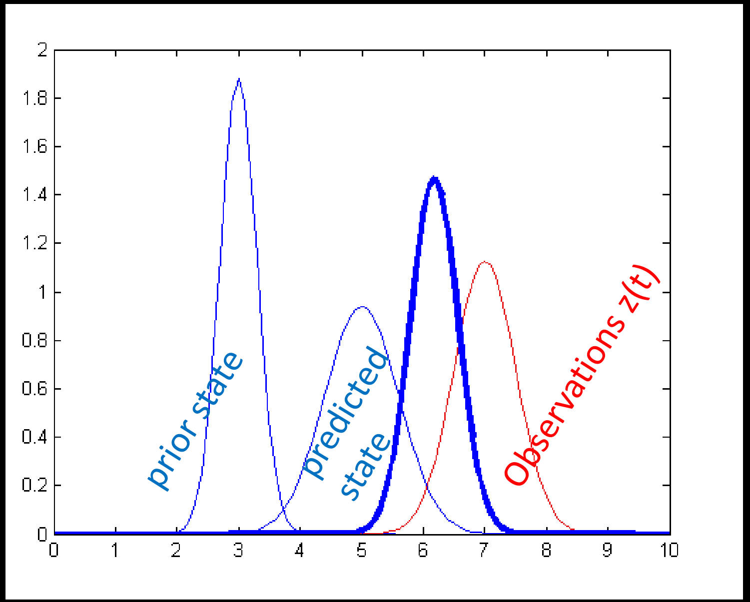
1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)\,C^T\,(C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(z(t) - C\,\mu_p(t))$
5. $\Sigma(t) = (I - K_{KF}\,C)\,\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

Steps 1–2: prediction

Steps 3–5: update

State estimate: $\mu(t)$
State uncertainty: $\Sigma(t)$
Process noise: Σ_u
Kalman filter gain: K_{KF}
Measurement noise: Σ_z





Kalman Filter

Kalman Filter ($\mu(t-1)$, $\Sigma(t-1)$, $u(t)$, $z(t)$)

1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)\,C^T\,(C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(z(t) - C\,\mu_p(t))$
5. $\Sigma(t) = (\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

prediction

update

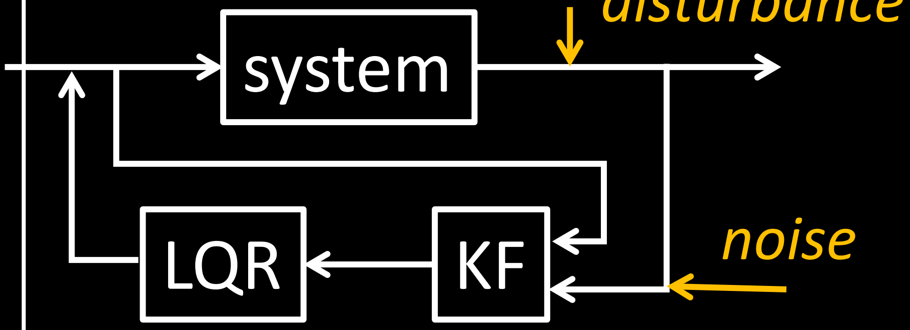
State estimate: $\mu(t)$

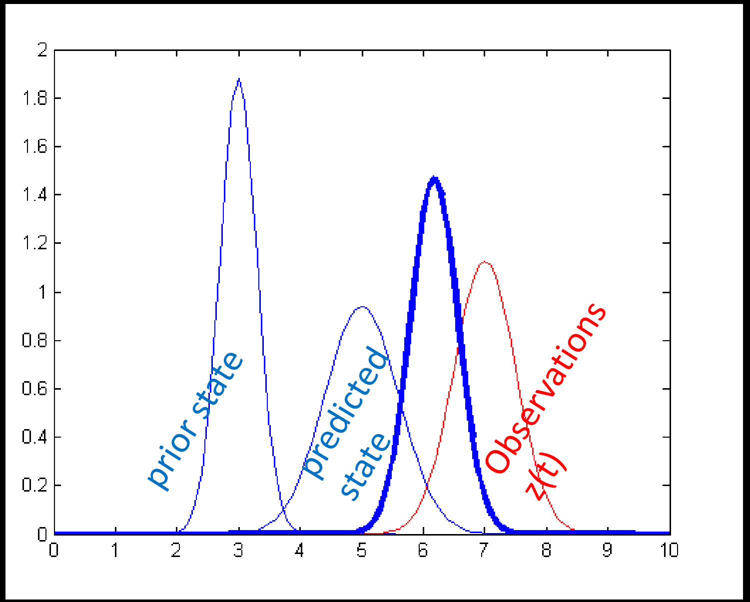
State uncertainty: $\Sigma(t)$

Process noise: Σ_u

Kalman filter gain: K_{KF}

Measurement noise: Σ_z





Example process and measurement noise covariance matrices:

$$\Sigma_u = \begin{bmatrix} \sigma_1{}^2 & 0 \\ 0 & \sigma_2{}^2 \end{bmatrix}, \Sigma_z = \sigma_3{}^2$$

Kalman Filter

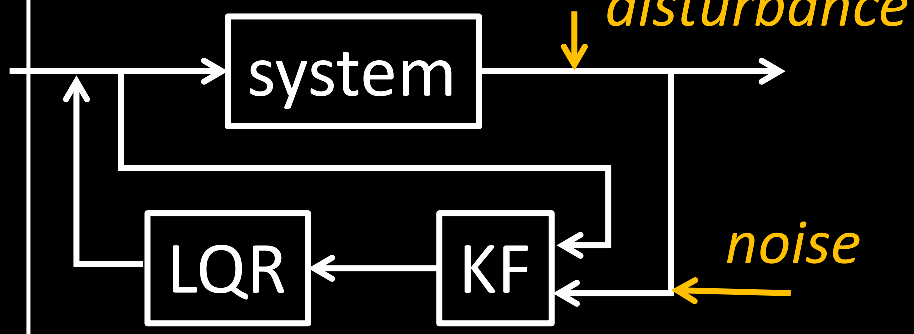
Kalman Filter ($\mu(t-1)$, $\Sigma(t-1)$, u(t), z(t))

1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)\,C^T\,(\,C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(\,z(t) - C\,\mu_p(t)\,)$
5. $\Sigma(t) = (\,\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

Steps 1–2: prediction

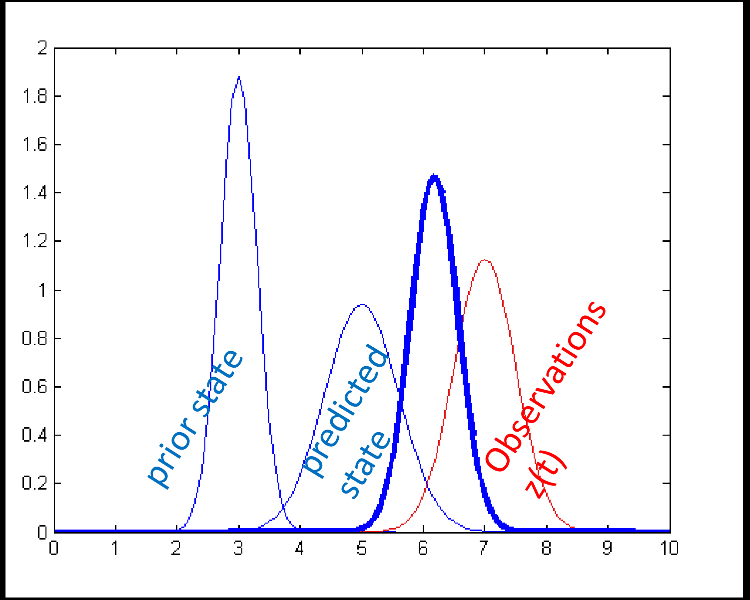
Steps 3–5: update

State estimate: $\mu(t)$
State uncertainty: $\Sigma(t)$
Process noise: Σ_u
Kalman filter gain: K_{KF}
Measurement noise: Σ_z



Example process and measurement noise covariance matrices:

$$\Sigma_u = \begin{bmatrix} {\sigma_1}^2 & 0 \\ 0 & {\sigma_2}^2 \end{bmatrix}, \Sigma_z = {\sigma_3}^2$$



Kalman Filter vs Bayes Filter

- Bayes Filter
- Kalman Filter uses the same idea, but uses Gaussian variables for posterior and prior beliefs to speed up computation.

Prior belief — input — observations

```
Bayes Filter(bel(x_{t-1}), u_t, z_t)
1. for all x(t) do
2.         bel‾(x(t)) = Σ (x(t-1)p(x(t)|u(t),x(t-1))bel(x(t-1))     Prediction step
3.         bel(x(t)) = α p(z(t)|x(t))bel‾(x(t))     Update step
4. end for
5. return bel(x_t)
```

$$\text{Bayes Filter}(bel(x_{t-1}), u_t, z_t)$$

$$\overline{bel}(x(t)) = \Sigma\, (x(t-1)p(x(t)|u(t),x(t-1))bel(x(t-1))$$

$$bel(x(t)) = \alpha\, p(z(t)|x(t))\overline{bel}(x(t))$$

Lab 6-8: PID control – Sensor Fusion - Stunt

- Task A: Position control
- Task B: Orientation control

Procedure

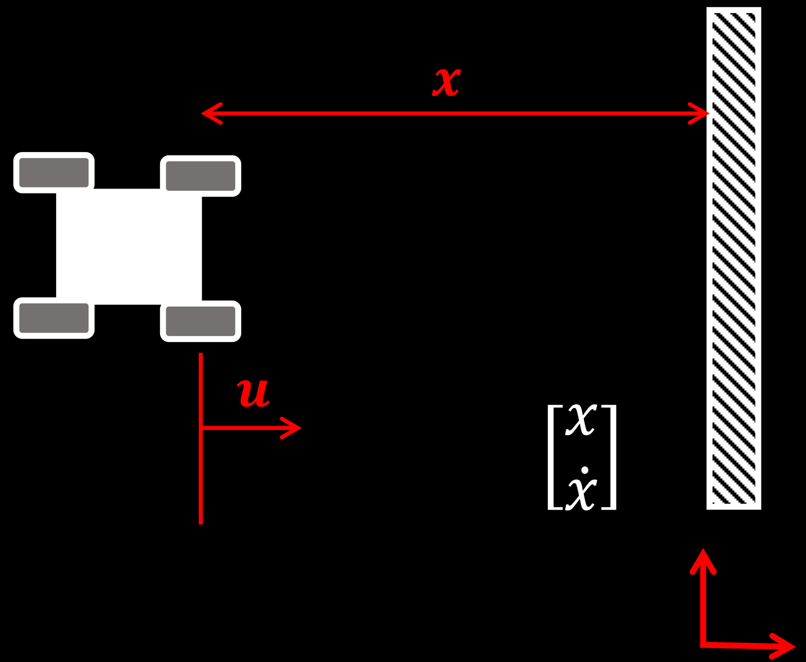
- Lab 6: Get basic PID to work , consider sampling time, start slow
- Lab 7: Sensor Fusion (model+ToF to get quick estimates of distance from the wall)
  - https://cei-lab.github.io/FastRobots-2023/Lab7.html
  - Do a step response with your robot and build your state space equations
  - Estimate covariance matrices for process and sensor noise
  - Try the Kalman Filter in Jupyter on your own data from lab 6
  - Implement the Kalman Filter on your robot
  - Great example: https://anyafp.github.io/ece4960/labs/lab7/

- Lab 8: Use KF and PID control to execute fast stunts

Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

What is d and m?



State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

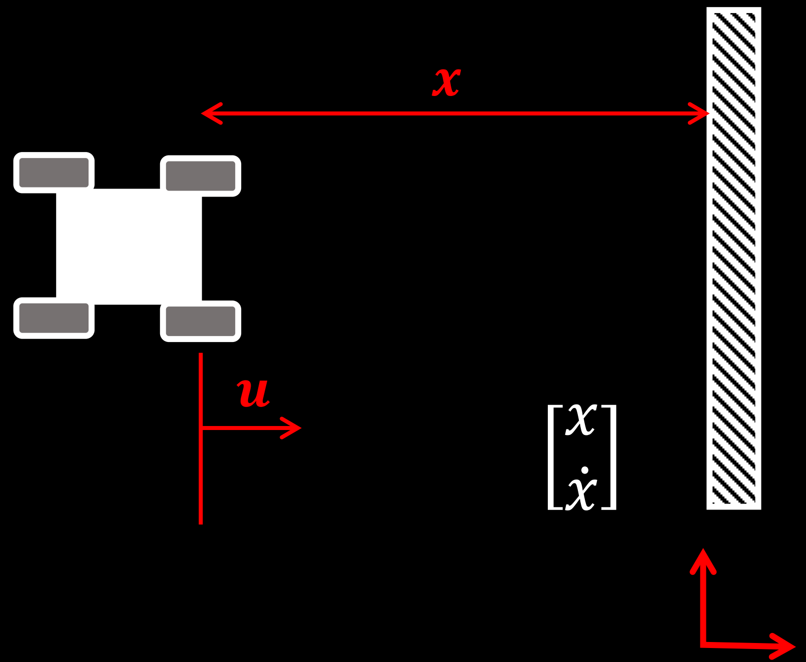
$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

What is d and m?

- At steady state (cst speed), we can find *d*
  - $0 = \frac{u}{m} - \frac{d}{m}\dot{x}$
  - $0 = \frac{u}{m} - \frac{d}{m}\dot{x} \quad \leftrightarrow \quad d = \frac{u}{\dot{x}}$



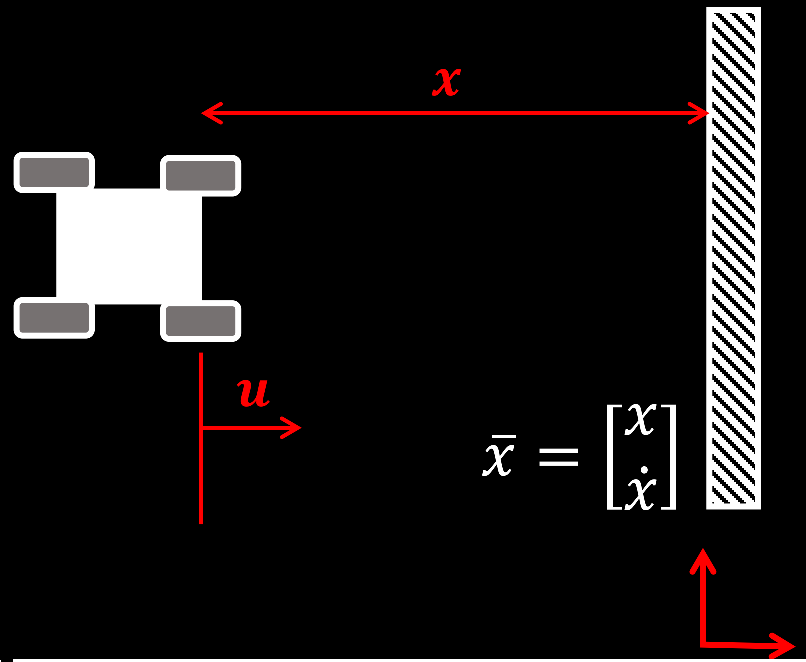
State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

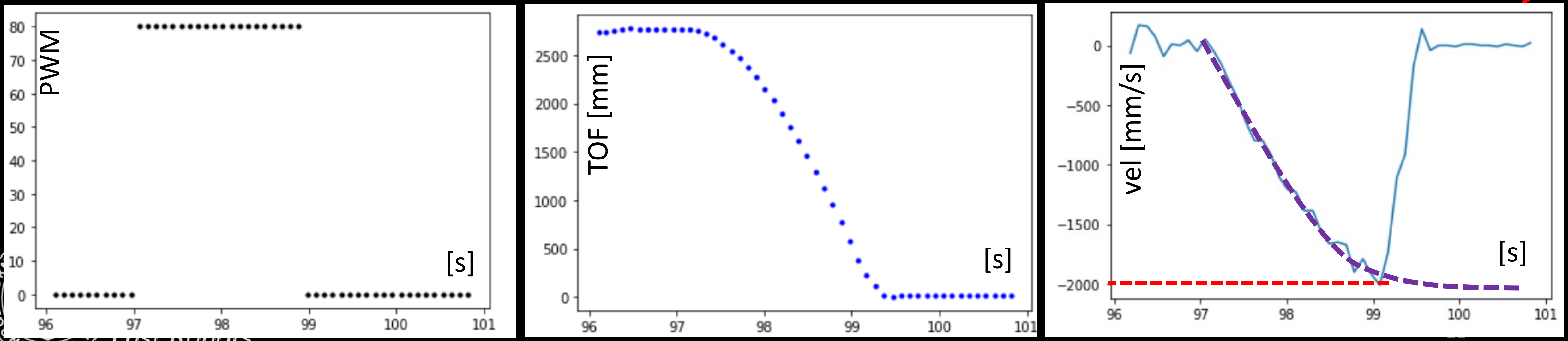
$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$



$$\bar{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

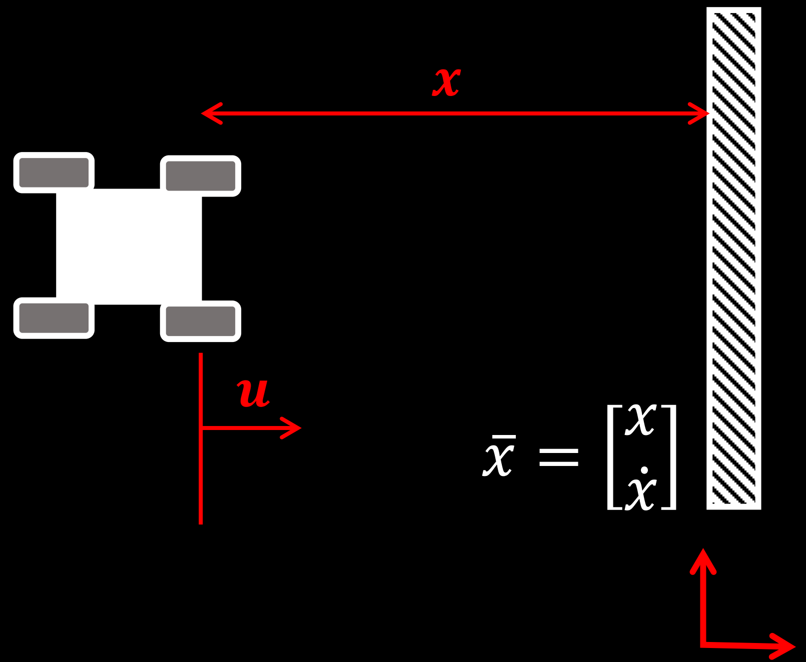
What is d and m?

- At steady state (cst speed), we can find *d*



Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$



What is d and m?

- At steady state (cst speed), we can find *d*
  - $0 = \frac{u}{m} - \frac{d}{m}\dot{x}$
  - $0 = \frac{u}{m} - \frac{d}{m}\dot{x} \quad \leftrightarrow \quad d = \frac{u}{\dot{x}}$
  - $d \approx \frac{1}{2000mm/s}$ *(Assume u=1 for now)*

State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

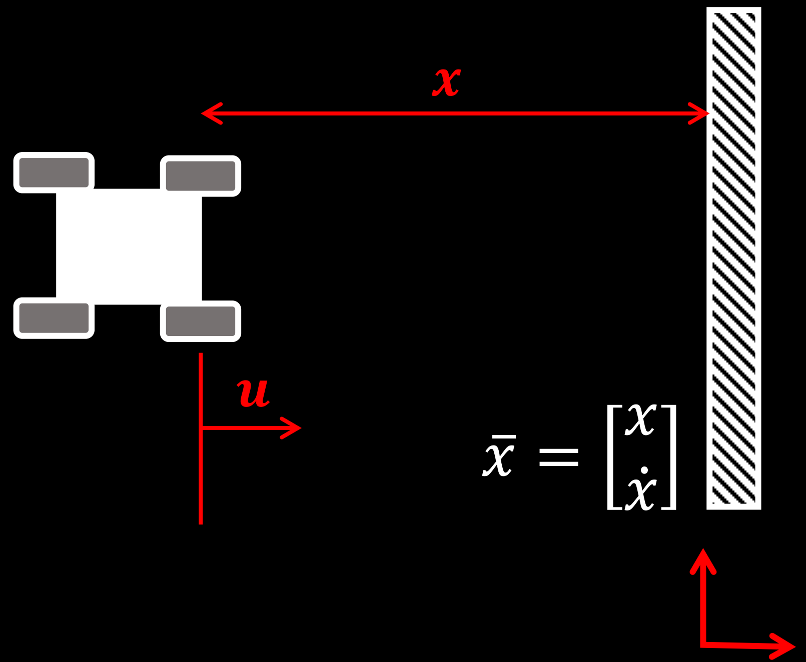
$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

1[st] order system:

$$\frac{dy(t)}{dt} + \frac{1}{\tau}y(t) = x(t)$$

Unit step response solution:

$$y(t) = 1 - e^{-\frac{t}{\tau}}$$



$$\bar{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

What is d and m?

- Use the rise time to determine m
  - $\dot{v} = \frac{u}{m} - \frac{d}{m}v$
  - $v = 1 - e^{-\frac{d}{m}t_{0.9}} \leftrightarrow 1 - v = e^{-\frac{d}{m}t_{0.9}}$
  - $\ln(1 - v) = -\frac{d}{m}t_{0.9}$
  - $m = \frac{-dt_{0.9}}{\ln(1-0.9)}$

State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

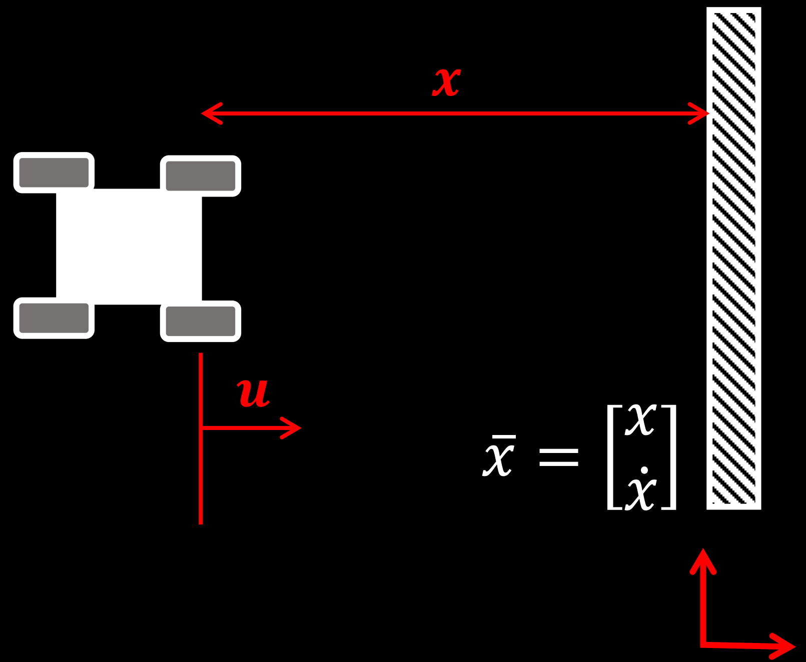
$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

1st order system:

$$\frac{dy(t)}{dt} + \frac{1}{\tau}y(t) = x(t)$$

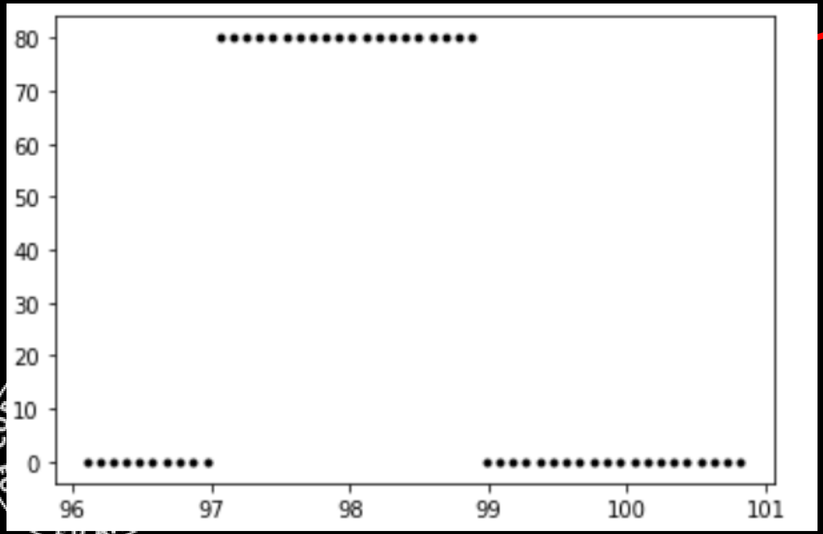
Unit step response solution:

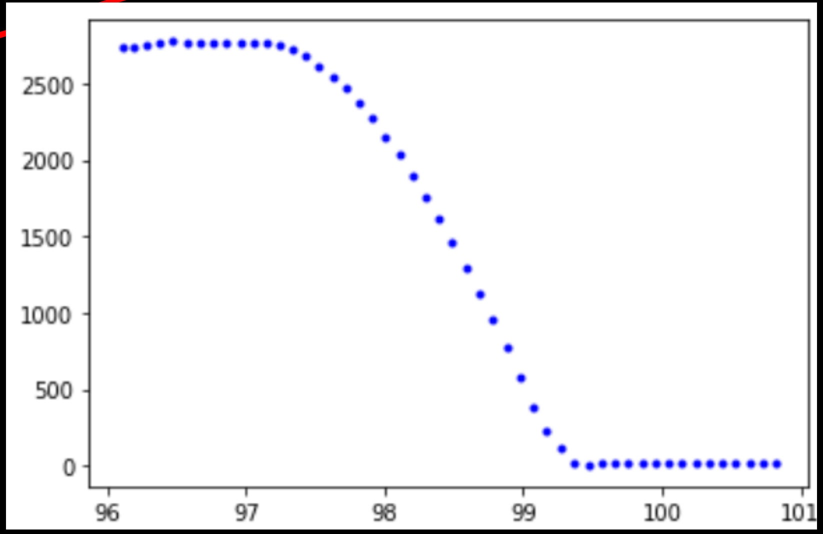
$$y(t) = 1 - e^{-\frac{t}{\tau}}$$

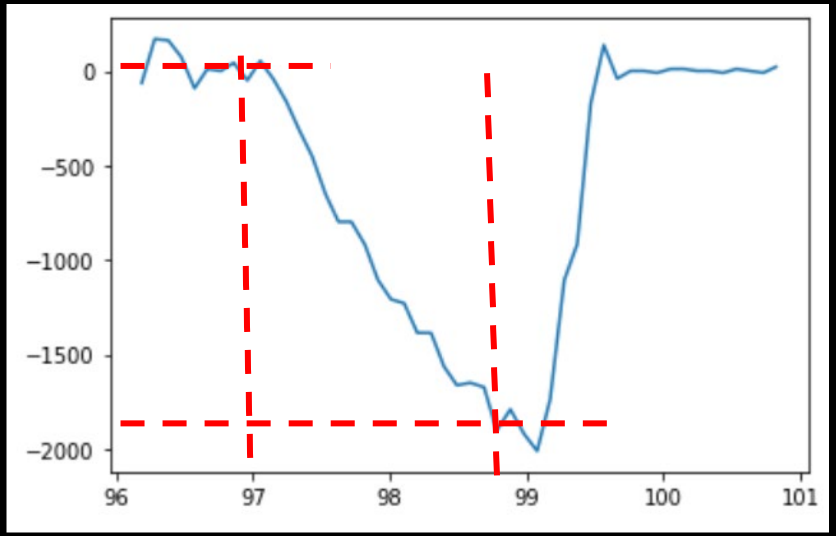


What is d and m?

- Use the rise time to determine m







Lab 7: Kalman Filter

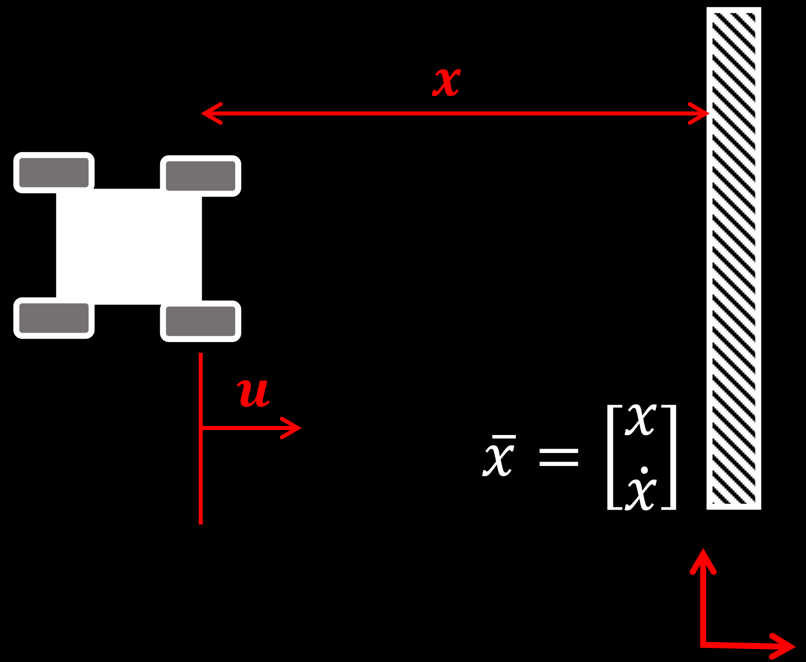
$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

1st order system:

$$\frac{dy(t)}{dt} + \frac{1}{\tau}y(t) = x(t)$$

Unit step response solution:

$$y(t) = 1 - e^{-\frac{t}{\tau}}$$



What is d and m?

- Use the 90% rise time to find m
  - $\dot{v} = \frac{u}{m} - \frac{d}{m}v$
  - $v = 1 - e^{-\frac{d}{m}t_{0.9}} \leftrightarrow 1 - v = e^{-\frac{d}{m}t_{0.9}}$
  - $\ln(1 - v) = -\frac{d}{m}t_{0.9}$
  - $m = \frac{-dt_{0.9}}{\ln(1-0.9)} = \frac{-0.0005 \cdot 1.9}{\ln(0.1)} = 4.1258 \cdot 10^{-4}$

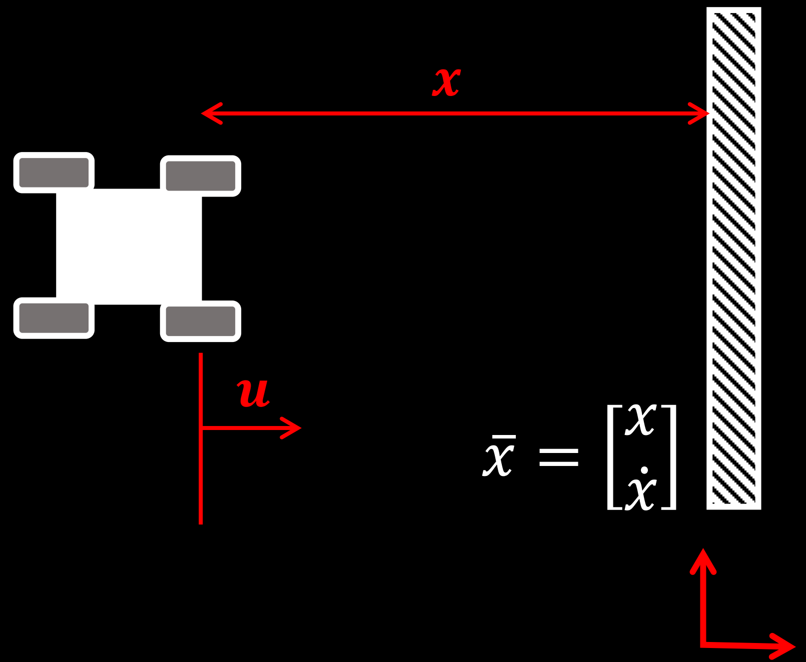
State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$
$$F = u - d\dot{x}$$
$$u - d\dot{x} = m\ddot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$



What is d and m?

- At steady state (cst speed), we can find *d*
  - $d = \frac{u}{\dot{x}} \approx 0.0005$ *(Assume u=1 for now)*
- We can use the rise time to find *m*
  - $m = \frac{-d t_{0.9}}{\ln(0.1)} \approx 4.1258 \cdot 10^{-4}$

State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

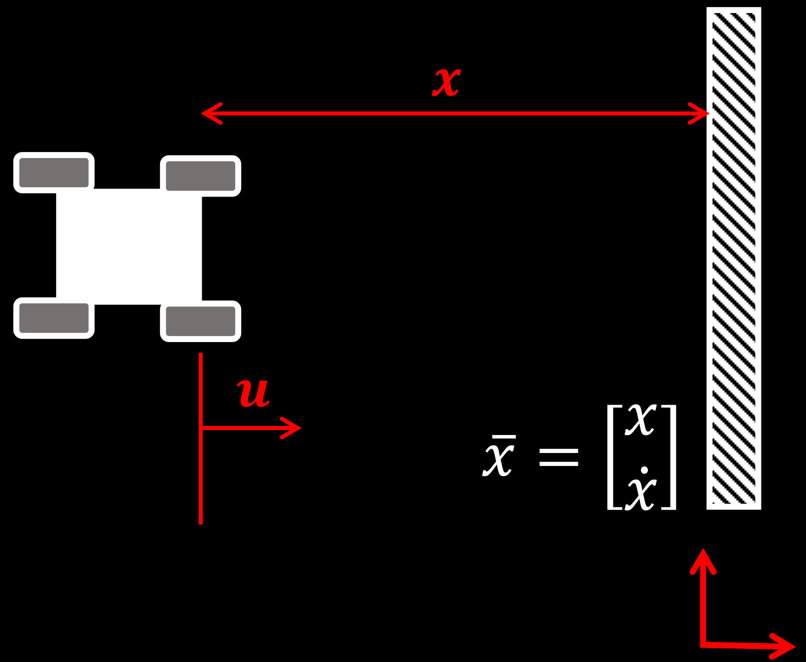
$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

- We have A, B, C, Σ_u, Σ_z
- Discretize the A and B matrices
  - x(n+1) = x(n) + dx
  - dx/dt = Ax+Bu ⇔ dx = dt (Ax + Bu)
  - x(n+1) = x(n) + dt (Ax(n) + Bu)
  - x(n+1) = (I + dt*A) x(n) + dt*B u

    (I + dt*A) = A_d, dt*B = B_d

  - dt is our sampling time (0.130s)
- Rescale from unity input to actual input



$$\bar{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

State space equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$C = \begin{bmatrix} -1 & 0 \end{bmatrix}$$

Lab 7: Kalman Filter

Implement the Kalman Filter

Next, determine measurement and process noise

Kalman Filter ($\mu(t-1)$, $\Sigma(t-1)$, u(t), z(t))

1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)\,C^T\,(\,C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(\,z(t) - C\,\mu_p(t)\,)$
5. $\Sigma(t) = (\,\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

```
def kf(mu,sigma,u,y):

    mu_p = A.dot(mu) + B.dot(u)
    sigma_p = A.dot(sigma.dot(A.transpose())) + Sigma_u

    sigma_m = C.dot(sigma_p.dot(C.transpose())) + Sigma_z
    kkf_gain = sigma_p.dot(C.transpose().dot(np.linalg.inv(sigma_m)))

    y_m = y-C.dot(mu_p)
    mu = mu_p + kkf_gain.dot(y_m)
    sigma=(np.eye(2)-kkf_gain.dot(C)).dot(sigma_p)

    return mu,sigma
```

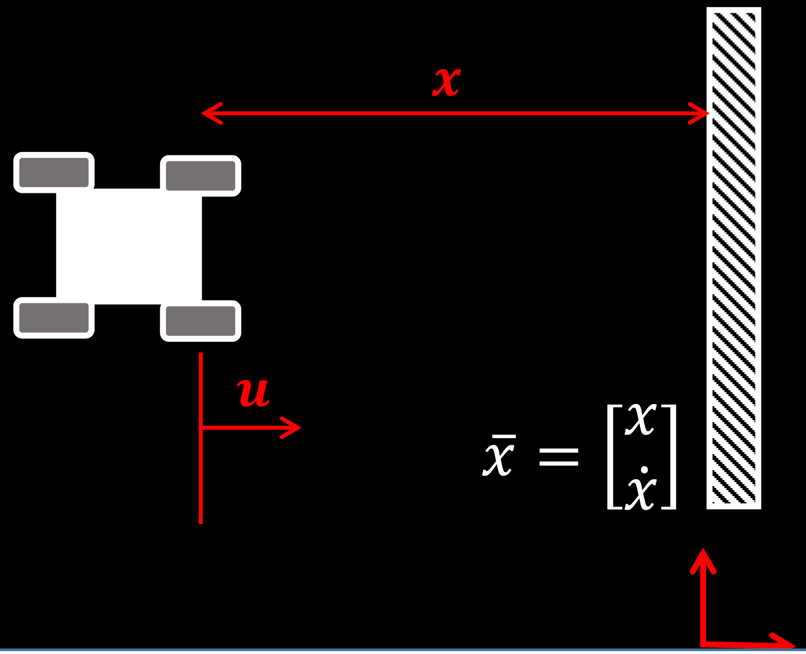
Lab 7: Kalman Filter

Implement the Kalman Filter

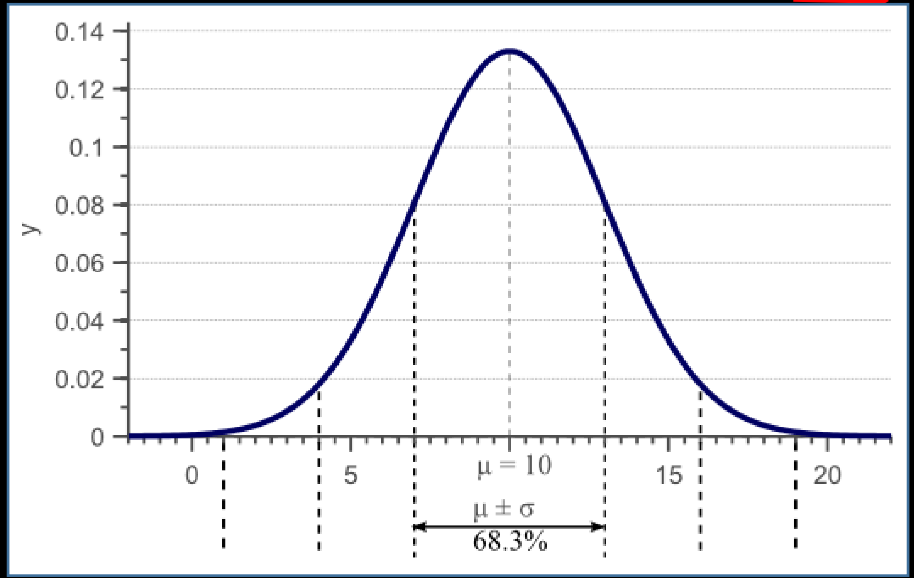
- Measurement noise
    - $\Sigma_z = [\sigma_3{}^2]$
    - $\sigma_3^2 = (20mm)^2$
- Process noise (dependent on sampling rate)

$$\Sigma_u = \begin{bmatrix} \sigma_1{}^2 & 0 \\ 0 & \sigma_2{}^2 \end{bmatrix}$$

- Trust in modeled position:
    - Pos_{stddev} after 1s: $\sqrt{10^2 \cdot \frac{1}{0.13}} = 27.7mm$
- Trust in modeled speed:
    - $Speed_{stddev}$ after 1s: $\sqrt{10^2 \cdot \frac{1}{0.13}} = 27.7mm/s$



1 sample per ~0.13s



Lab 7: Kalman Filter

Implement the Kalman Filter

Kalman Filter ($\mu(t-1)$, $\Sigma(t-1)$, $u(t)$, $z(t)$)

1. $\mu_p(t) = A\,\mu(t-1) + B\,u(t)$
2. $\Sigma_p(t) = A\,\Sigma(t-1)\,A^T + \Sigma_u$
3. $K_{KF} = \Sigma_p(t)\,C^T\,(\,C\,\Sigma_p(t)\,C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}\,(\,z(t) - C\,\mu_p(t)\,)$
5. $\Sigma(t) = (\,\mathbf{I} - K_{KF}\,C)\,\Sigma_p(t)$
6. Return $\mu(t)$ and $\Sigma(t)$

Finally, determine your initial state mean and covariance

$\mu(t-1)$

$\Sigma(t-1)$

Play video!!

```
def kf(mu,sigma,u,y):

    mu_p = A.dot(mu) + B.dot(u)
    sigma_p = A.dot(sigma.dot(A.transpose())) + Sigma_u

    sigma_m = C.dot(sigma_p.dot(C.transpose())) + Sigma_z
    kkf_gain = sigma_p.dot(C.transpose().dot(np.linalg.inv(sigma_m)))

    y_m = y-C.dot(mu_p)
    mu = mu_p + kkf_gain.dot(y_m)
    sigma=(np.eye(2)-kkf_gain.dot(C)).dot(sigma_p)

    return mu,sigma
```

Lab 7: Kalman Filter