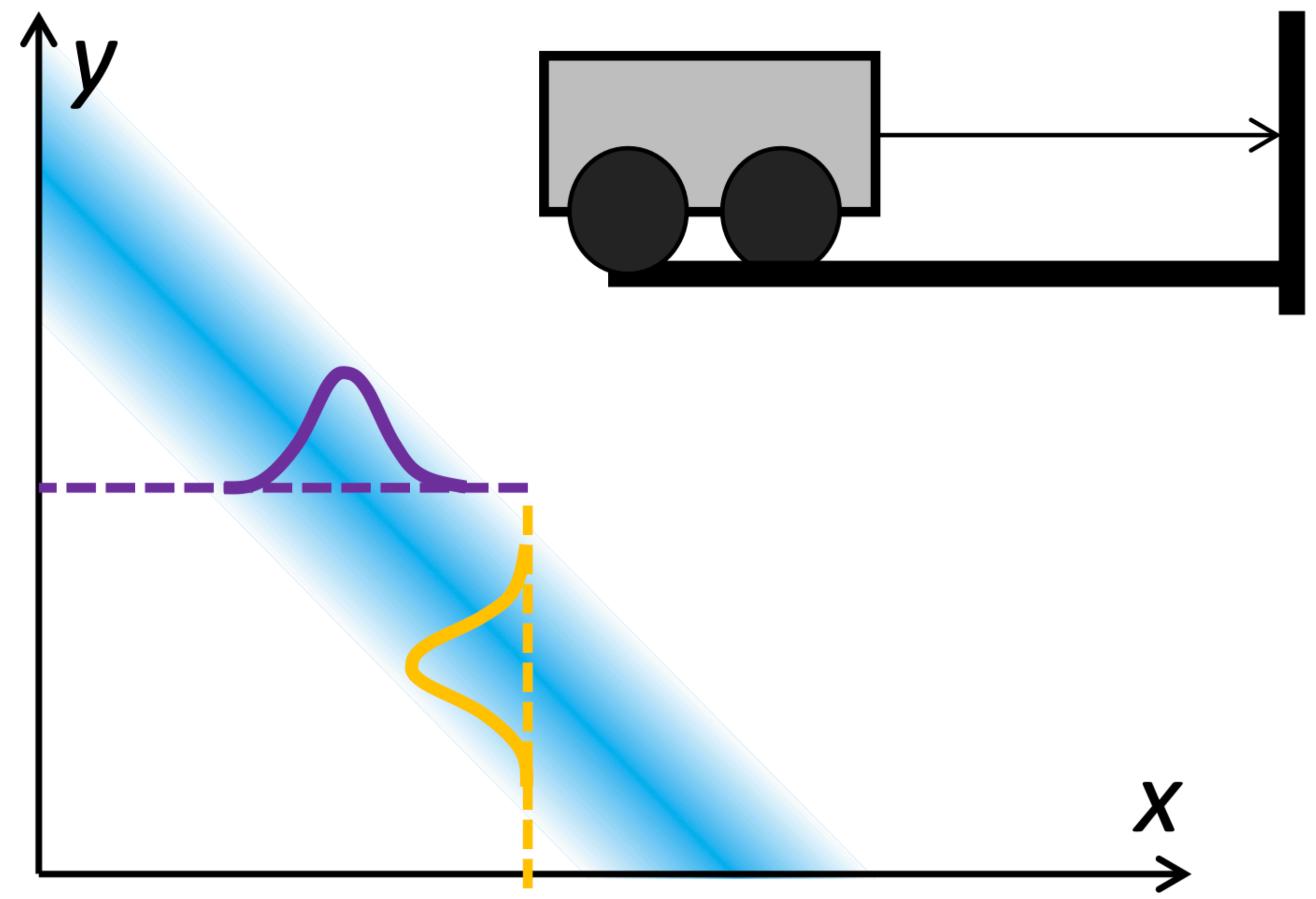
Observability

Fast Robots, ECE4160/5160, MAE 4190/5190

E. Farrell Helbling, 3/6/25

Class Action Items

- Lab 5 check in: how is everything going?
- I am hosting additional open hours tomorrow 8:30-11am and Sunday 6-8pm (moved from the original 11am-1pm), multiple requests for evening hours
- Tuesday's class: how are we feeling about probability?



Linear Systems — where are we?

- Linear systems review
- Eigenvectors and eigenvalues
- Stability
- Discrete time systems
- Linearizing nonlinear systems
- Controllability
- LQR control
- Observability

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

These should look familiar from:

- MATH2940 Linear Algebra
- ECE3250 Signals and Systems
- ECE5210 Theory of Linear Systems
- MAE3260 System Dynamics
- and many others…

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

Review of the Review

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

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

- Linear Transform: $AT = TD$
- Solution: $e^{At} = e^{TDT^{-1}t}$
- 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: $x(k+1) = \tilde{A}x(k)$, where $\tilde{A} = e^{A\Delta t}$
- Stability in discrete time: $\tilde{\lambda}^n = R^n e^{in\theta}$, stable iff $R < 1$
- Nonlinear systems: $\dot{x} = f(x)$
- Linearization: $\left.\frac{Df}{Dx}\right|_{\bar{x}}$
- Controllability: $\dot{x} = (A - BK)x$ `>>rank(ctrb(A,B))`
- Reachability
- Controllability Gramian
- Pole Placement `>>place(A,B,poles)`
- Optimal Control (LQR) `>>LQR(A,B,Q,R)`

Controllability

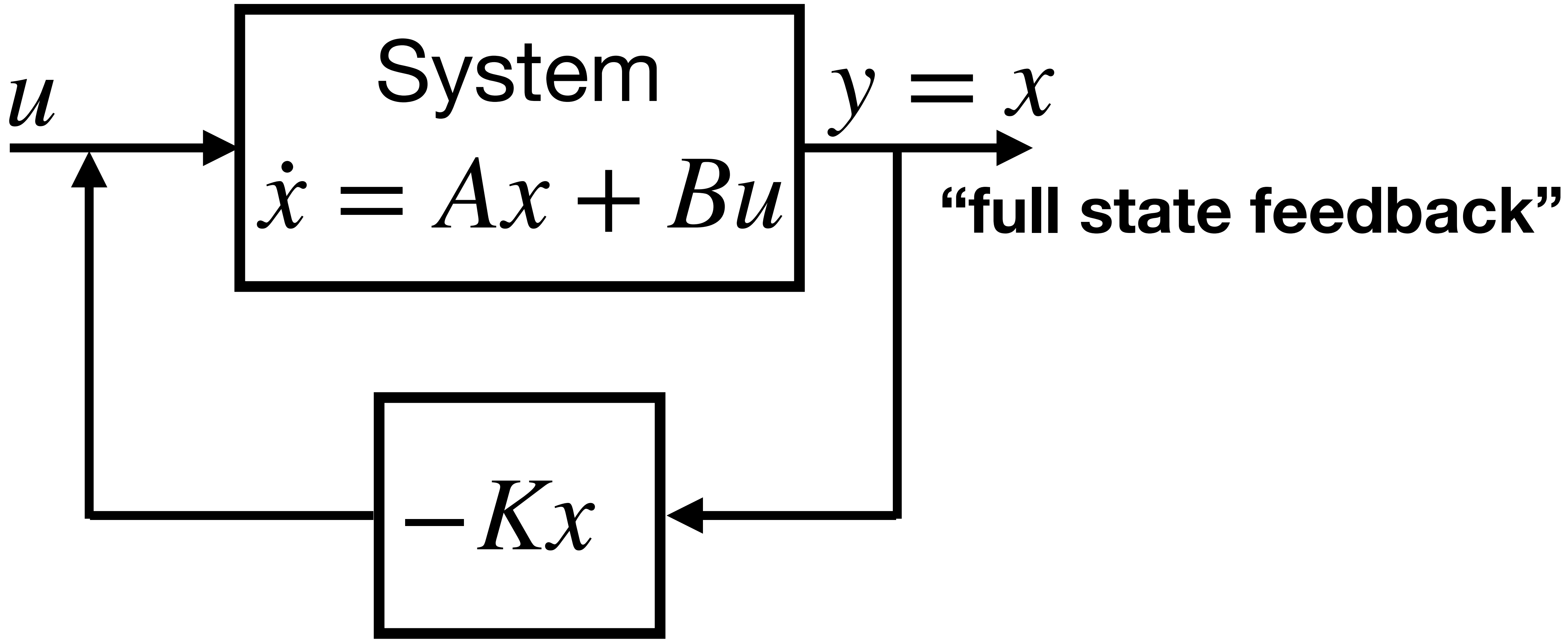
$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n$$

$$\dot{x} = Ax - BKx \qquad A \in \mathbb{R}^{n \times n}$$

$$\dot{x} = \underline{(A - BK)}x \qquad u \in \mathbb{R}^q$$

New dynamics $\qquad B \in \mathbb{R}^{n \times q}$

- Is the system controllable?
  - A system is controllable if you can steer your state x anywhere you want in $\mathbb{R}^n$
  - Matlab `>>rank(ctrb(A,B))`
- How do we design the control law, u?



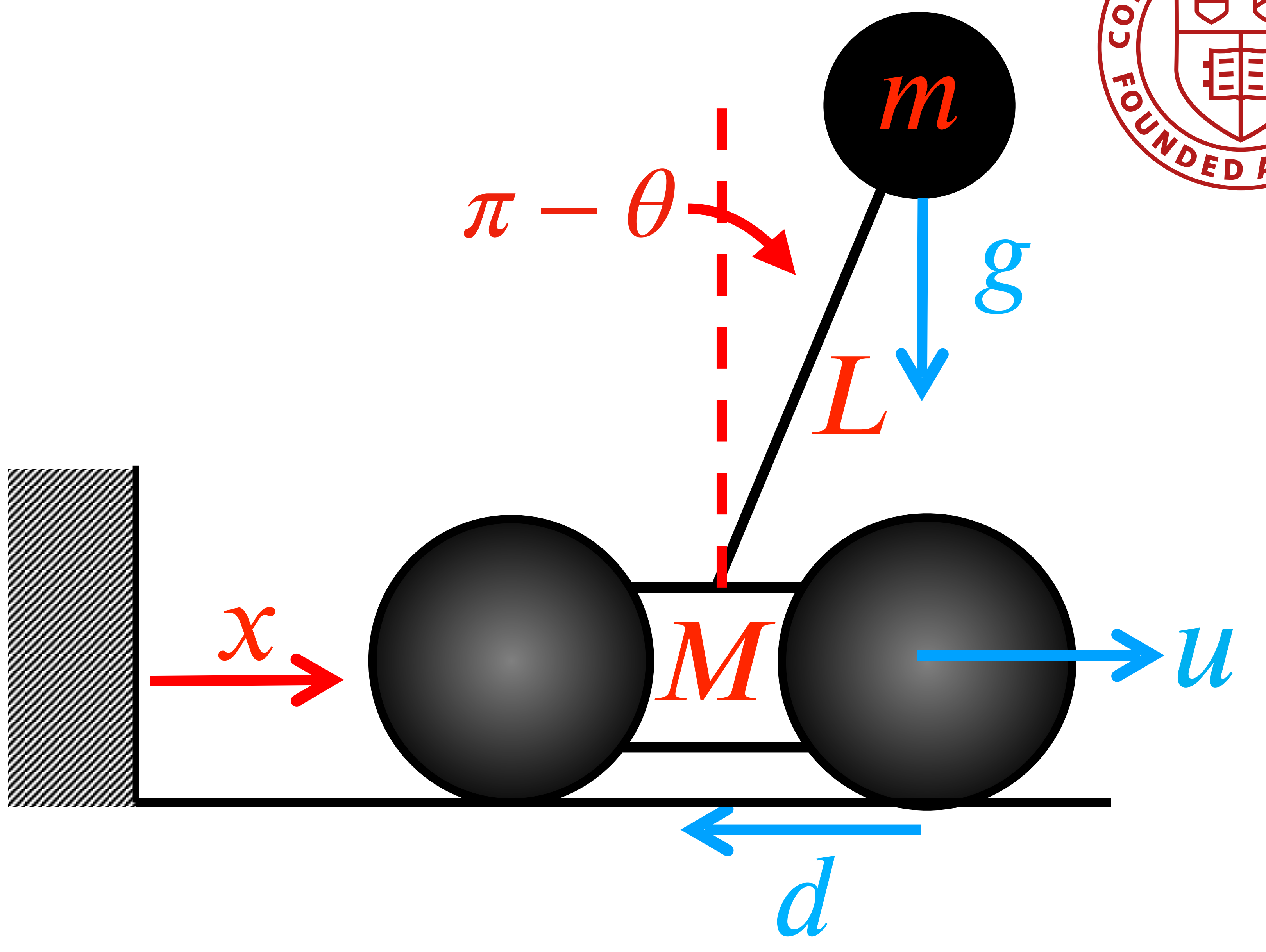
A linear controller (K matrix) can be optimal for linear systems!

Linear Quadratic Regulator

- What are the optimal eigenvalues for our system?
  - Tradeoff performance and control effort
- Define cost function: $\int_0^\infty (x^T Q x + u^T R u) dt$
  - $Q = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 10 & \\ & & & 100 \end{bmatrix}$ cost of my state being away from setpoint
  - $R = 0.001$ cost of input energy
  - Solved using the Ricatti Equation (compute expensive $O(n^3)$)
- Matlab `>>K = lqr(A,B,Q,R)`

Inverted pendulum on a cart

The controller works!



Caveats:

- In simulation
- Pratical issues:
  - Imperfect models
  - Nonlinear parts: deadband, saturation, etc.
  - Partial state feedback

Linear Systems — where are we?

- Linear systems review
- Eigenvectors and eigenvalues
- Stability
- Discrete time systems
- Linearizing nonlinear systems
- Controllability
- LQR control
- Observability

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

These should look familiar from:

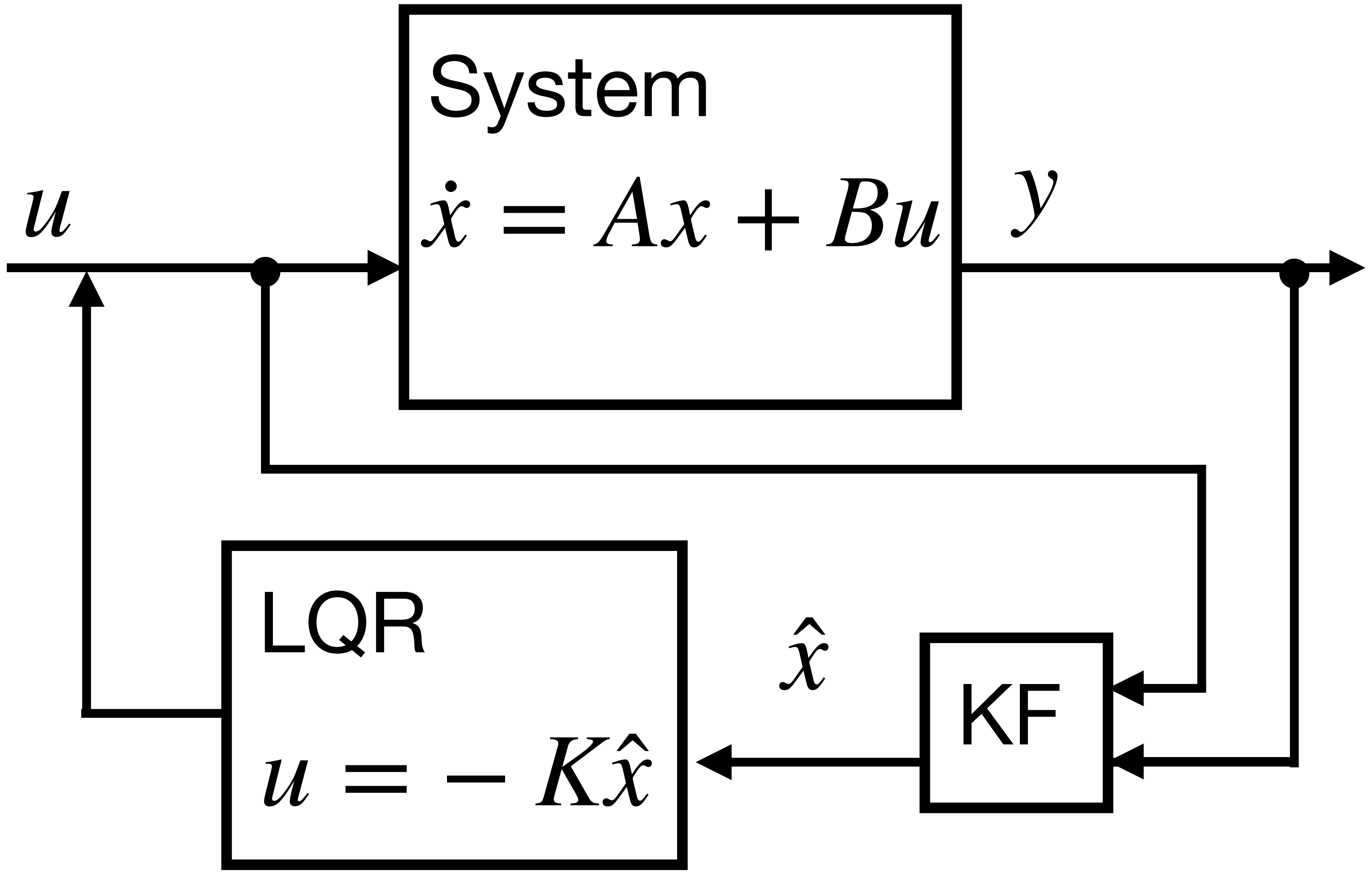
- MATH2940 Linear Algebra
- ECE3250 Signals and Systems
- ECE5210 Theory of Linear Systems
- MAE3260 System Dynamics
- and many others…

Observability

Observability

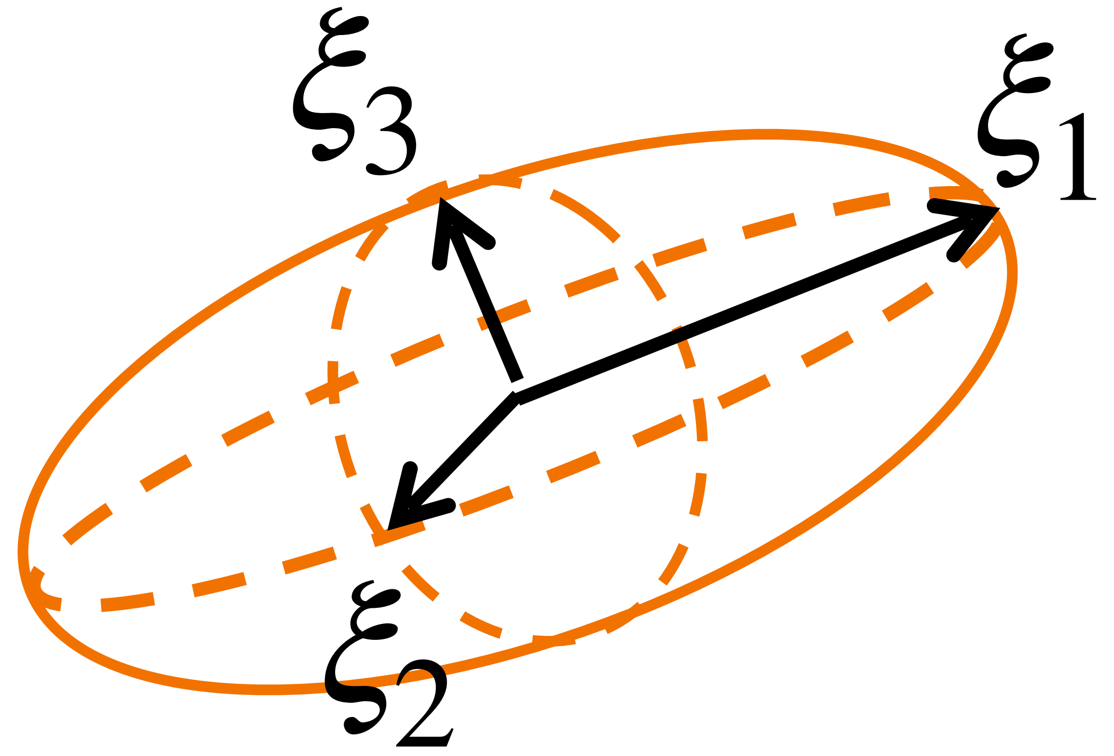
$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n$$
$$u = -Kx$$
$$\dot{x} = (A - BK)x$$

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



Observability

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



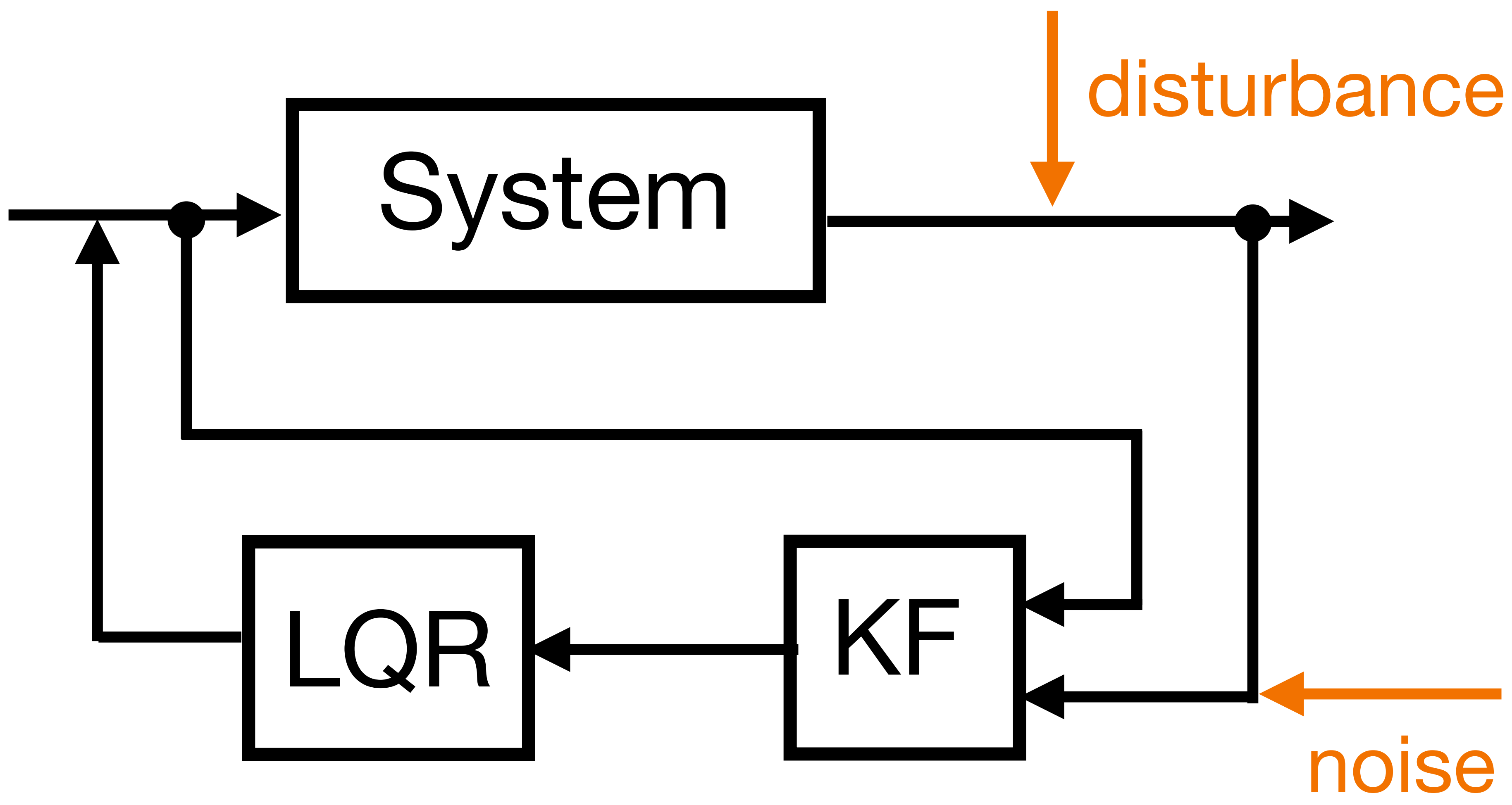
$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n$$
$$y = Cx \qquad u \in \mathbb{R}^q$$
$$y \in \mathbb{R}^p$$

$$\mathbb{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}$$

`>>rank(ctrb(A,B))`

- Reachability

- Observable iff rank($\mathcal{O}$) = n
  - `>>rank(obsv(A,C))`
- Iff a system is observable, we can estimate x from y. We can find the best estimates using the observability gramian
  - `>>[U, S, V] = svd(`$\mathcal{O}$`)`

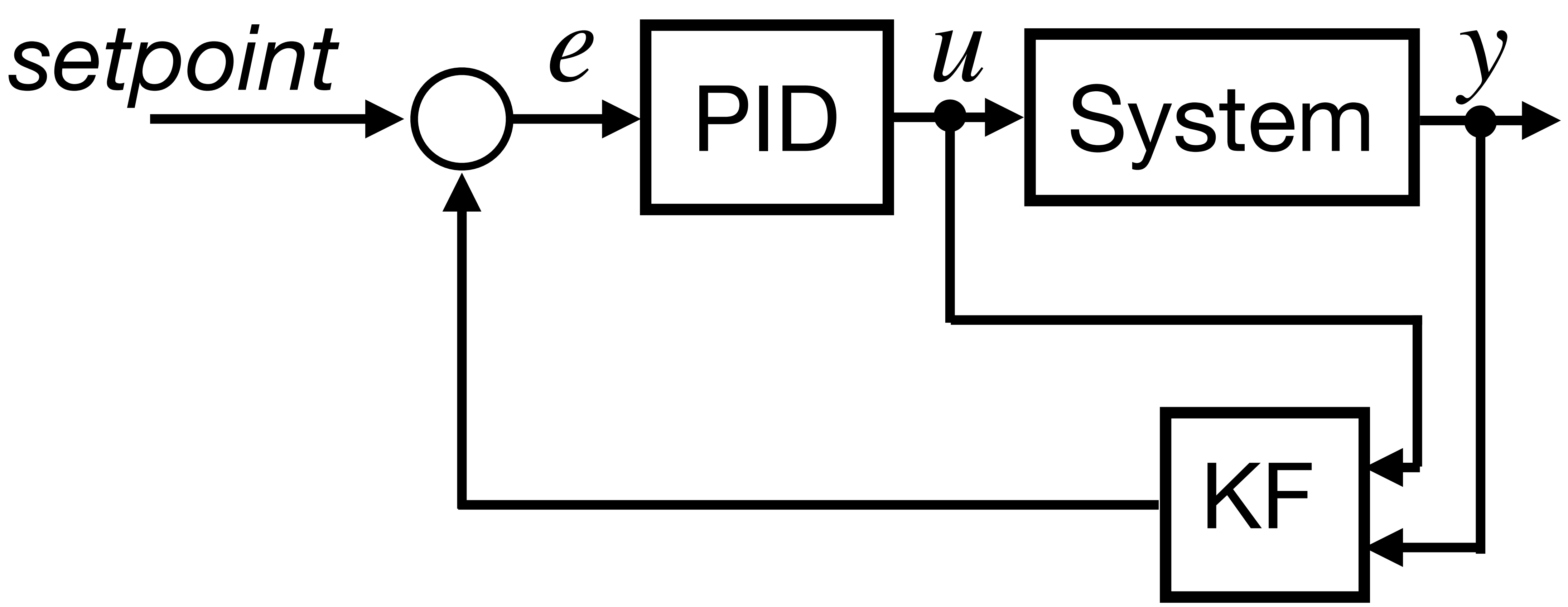


Kalman Filter

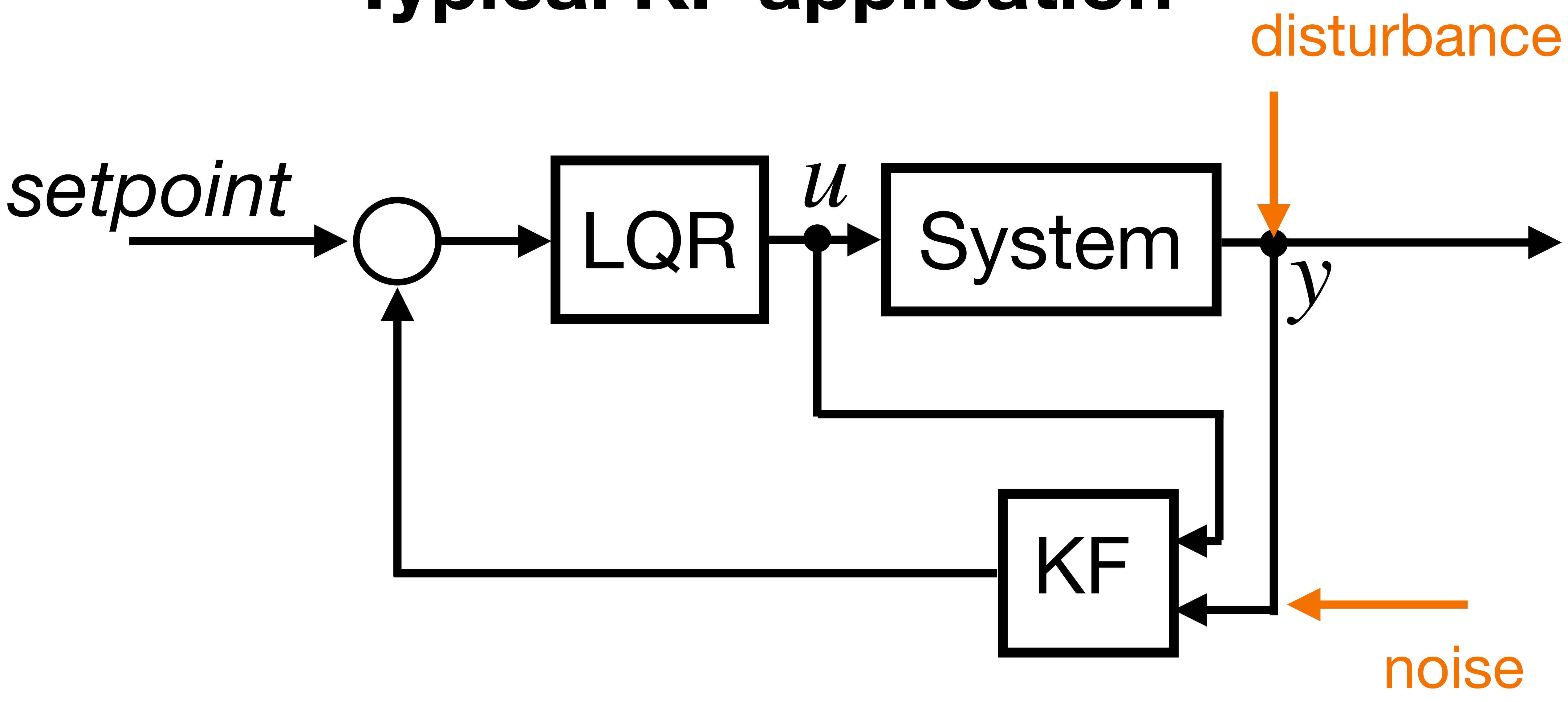
Why sensor fusion?

- Partial state feedback
- Bad sensors
- Imperfect model
- Slow feedback

KF with PID

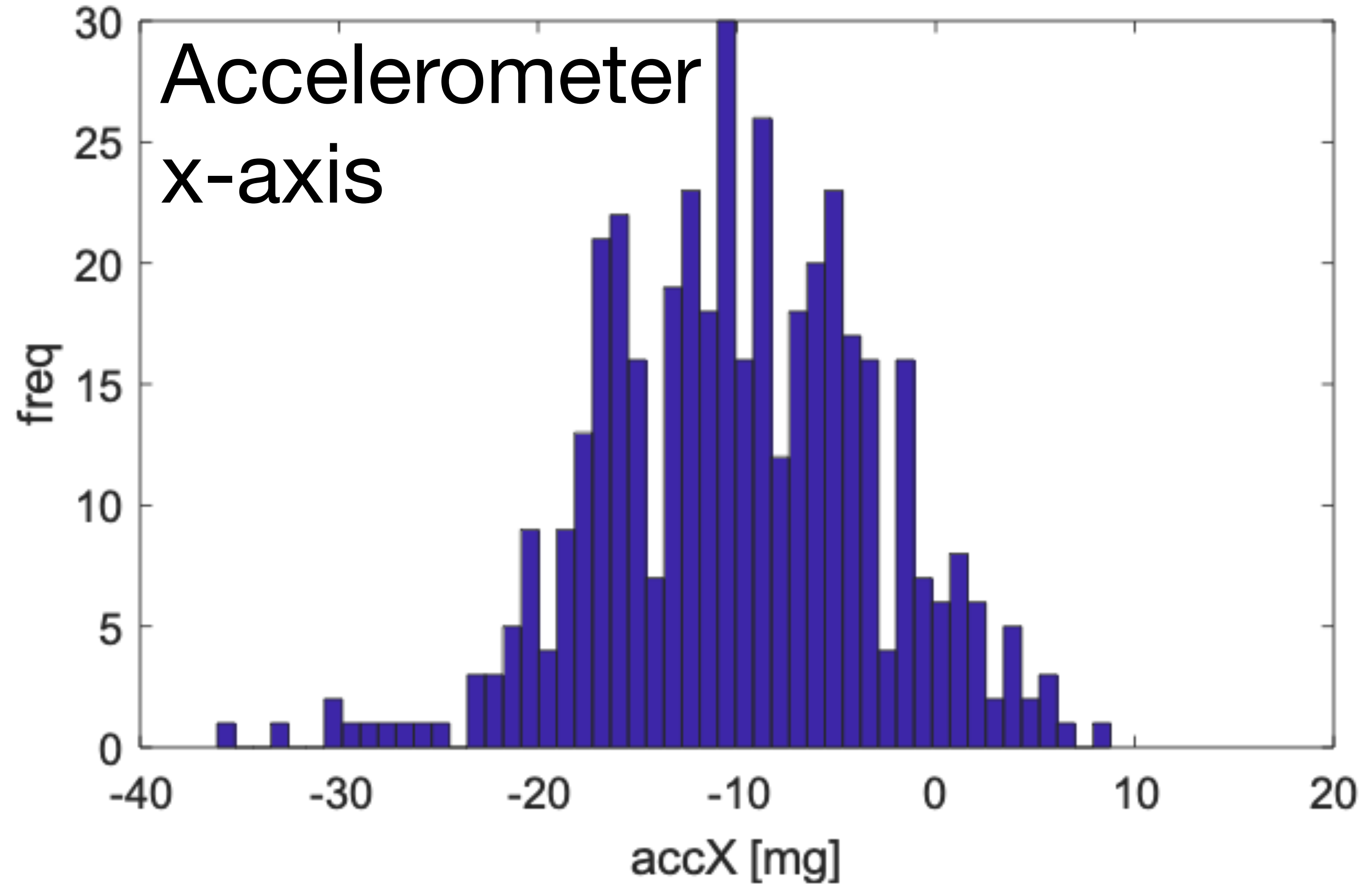


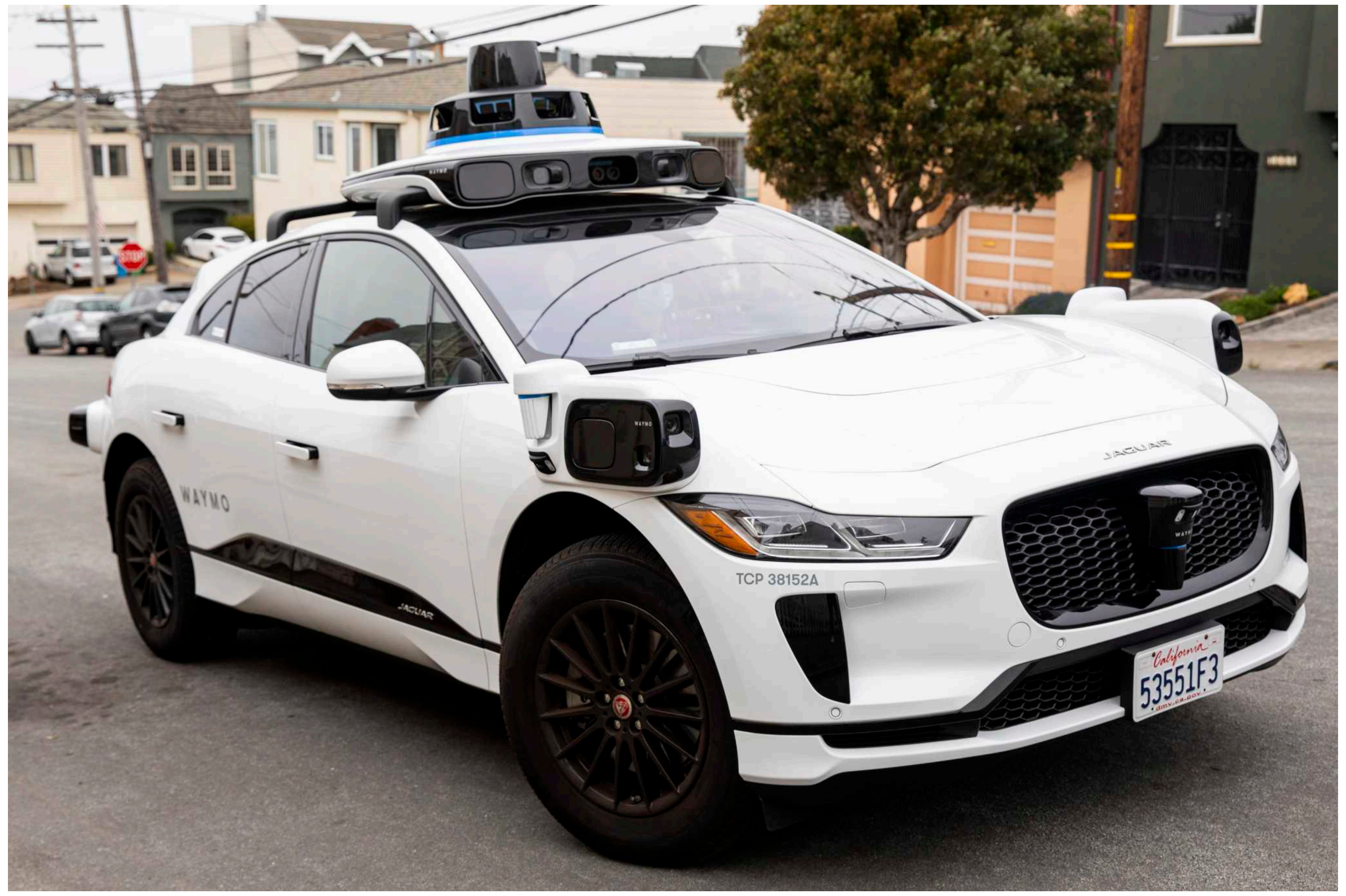
Typical KF application



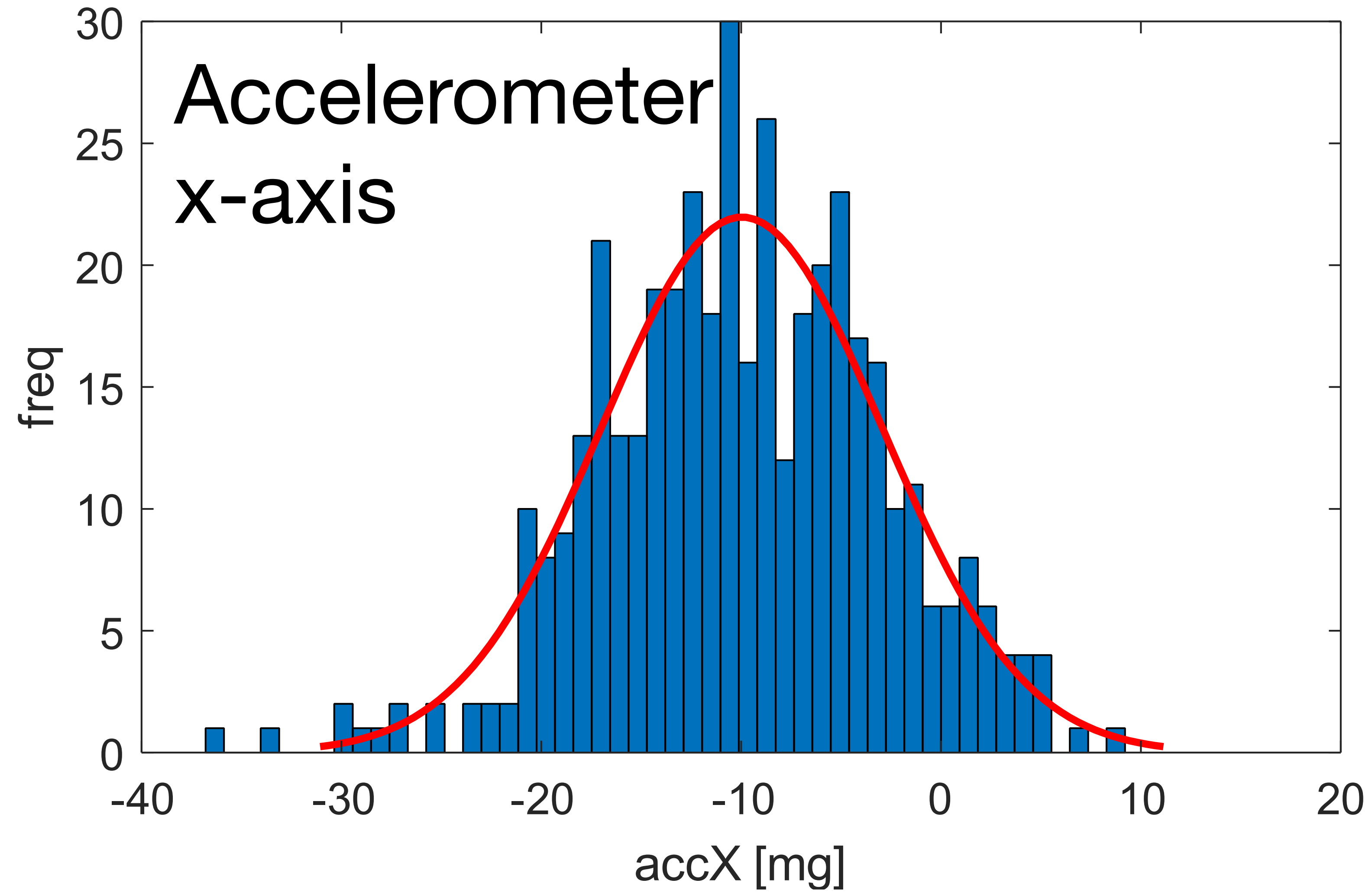
Probabilistic Robotics

- Sources of uncertainty
  - Measurements
  - Actions
  - Models
  - States



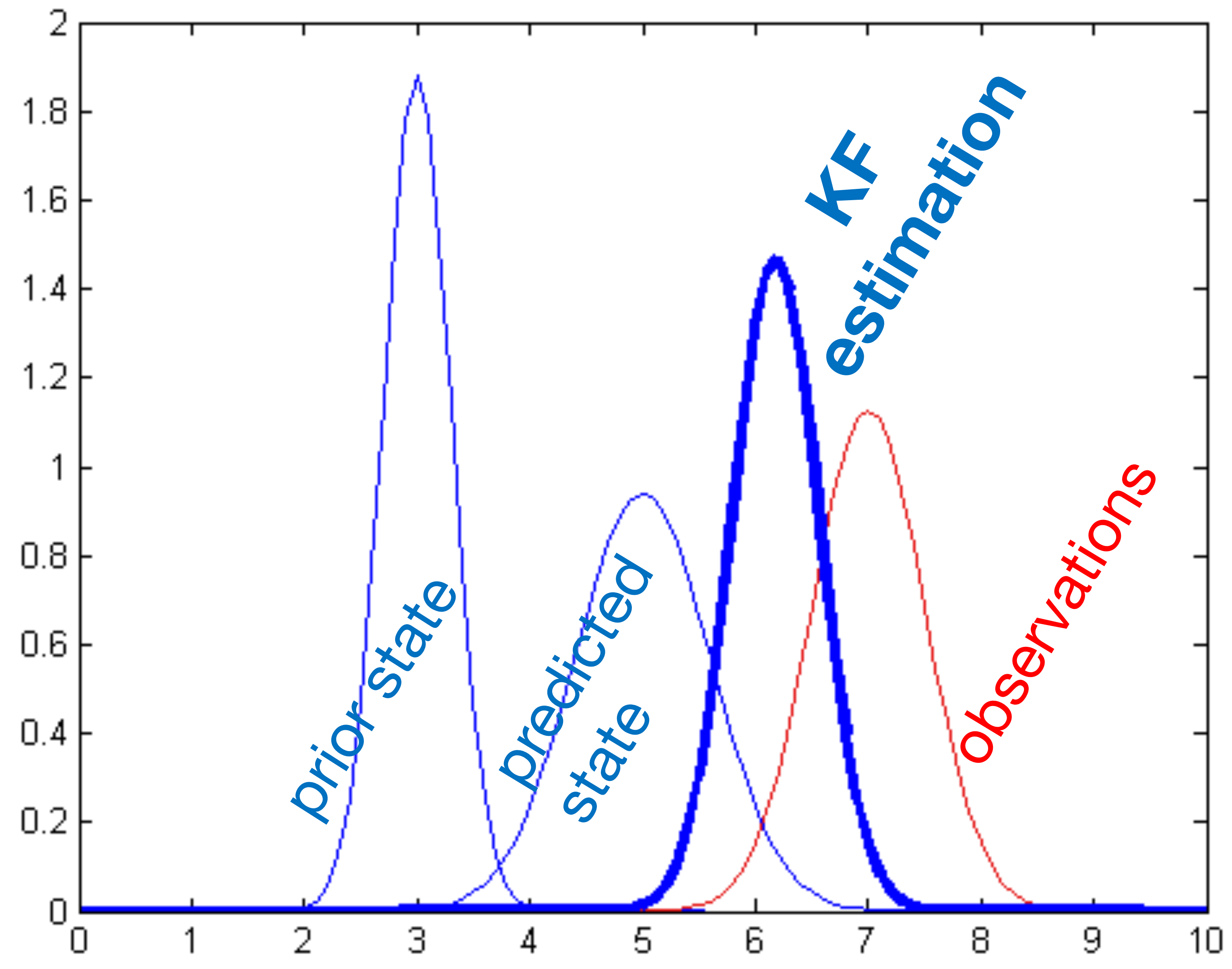


- Gaussian distributions
  - $[\mu \pm \sigma]$
  - Symmetric
  - Unimodal
  - Sum to "unity"



Kalman Filter

- Incorporate uncertainty to get better estimates based on both inputs and observations
  - Assume that posterior and prior belief are Gaussian variables



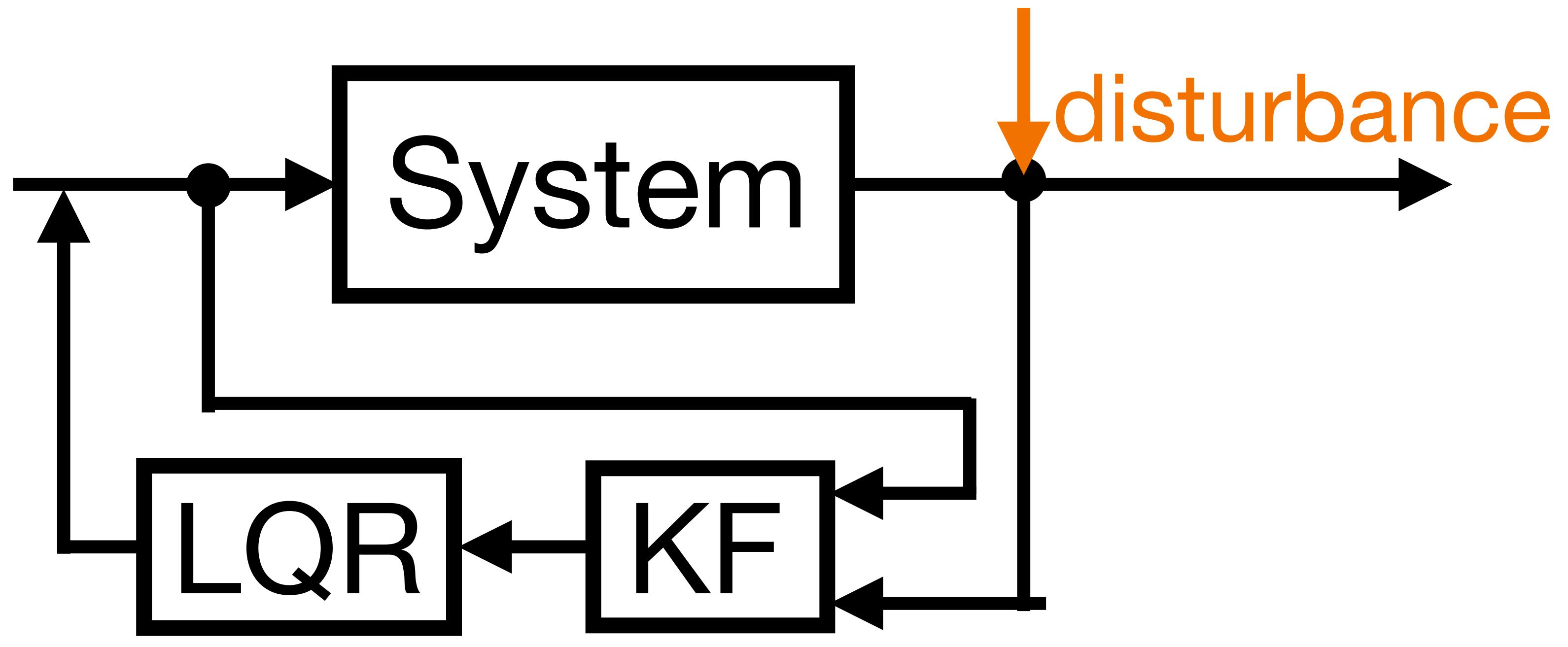
Kalman Filter

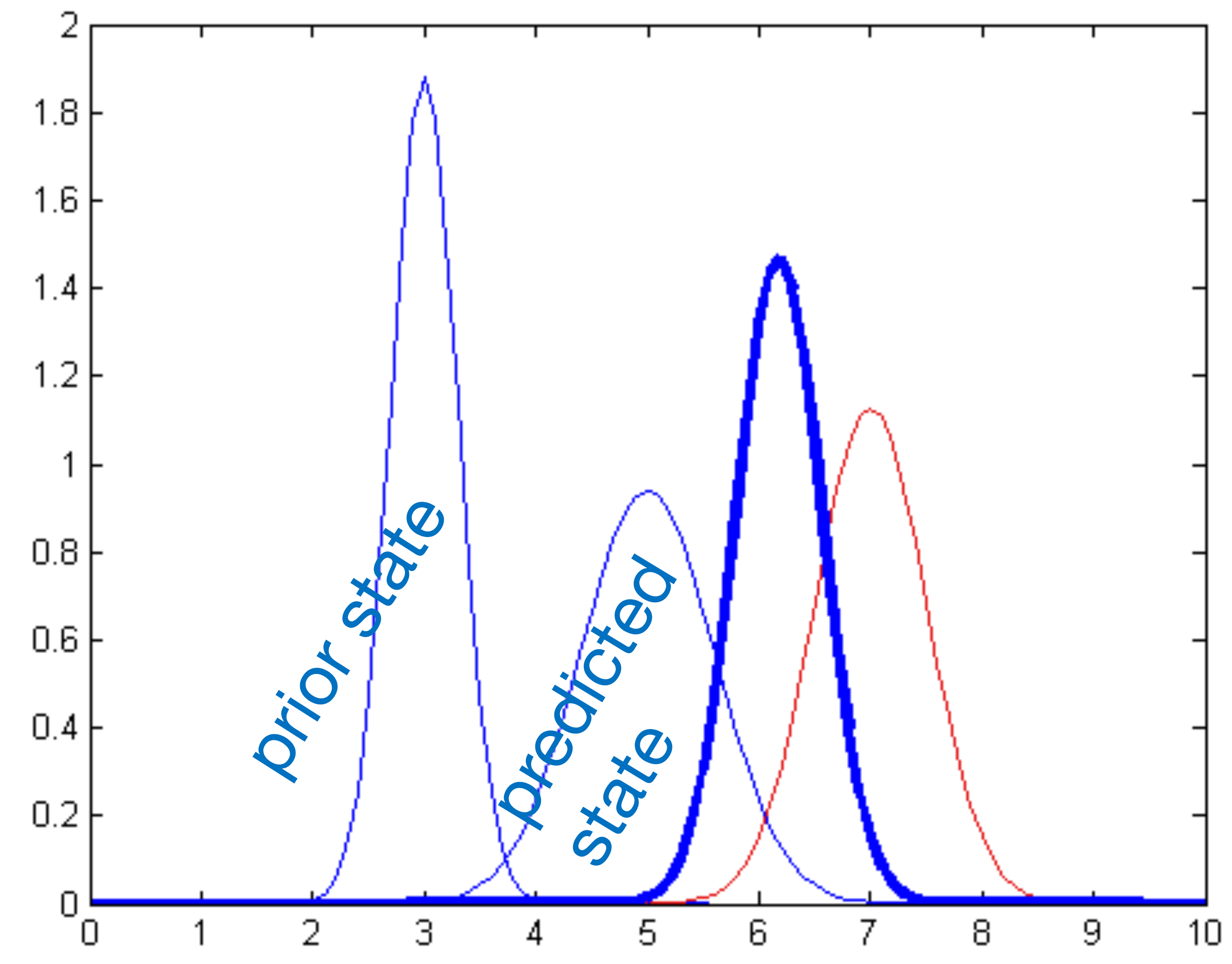
State estimate: $\mu(t)$

State uncertainty: $\Sigma(t)$

Process noise: Σ_u

- Assume that posterior and prior belief are Gaussian variables





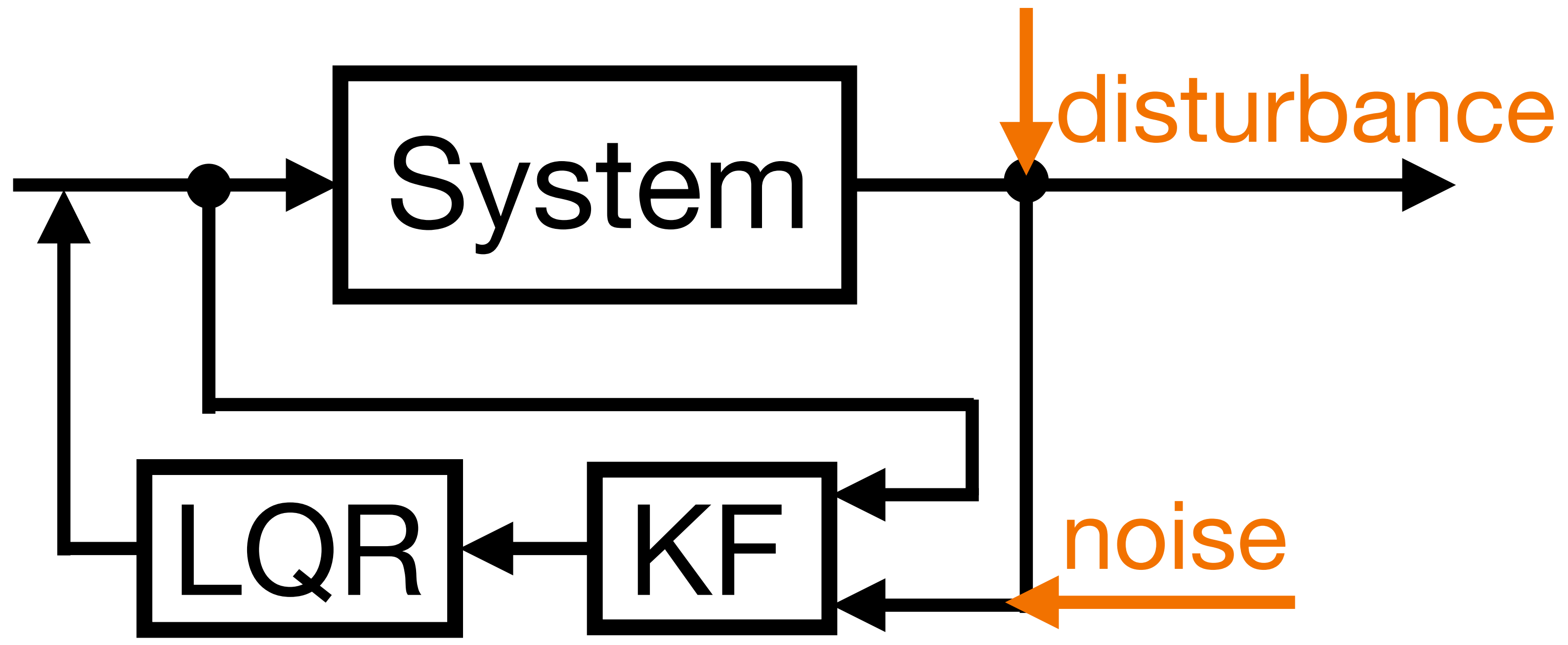
Kalman Filter

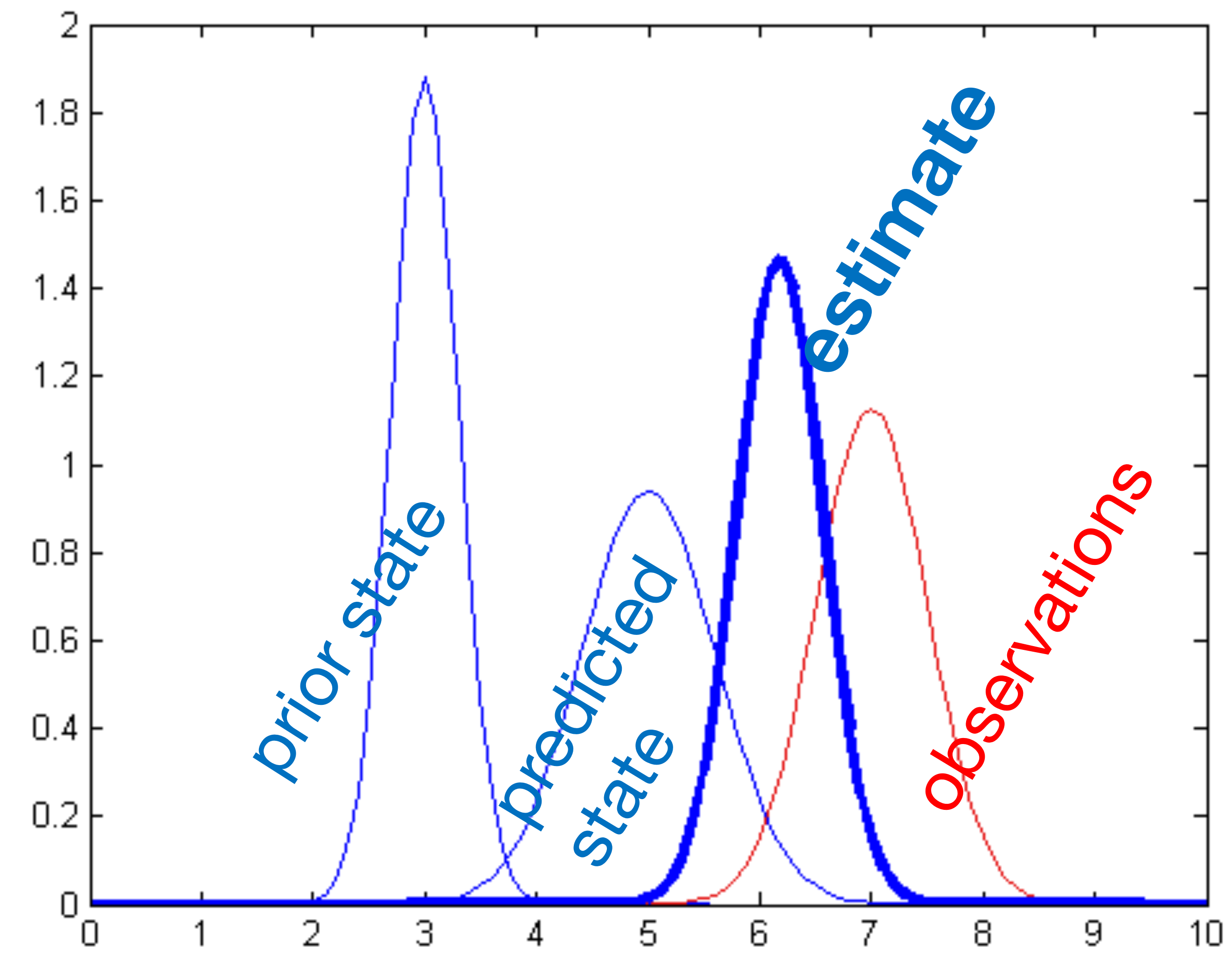
State estimate: $\mu(t)$

State uncertainty: $\Sigma(t)$

Process noise: Σ_u

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





Kalman Filter

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

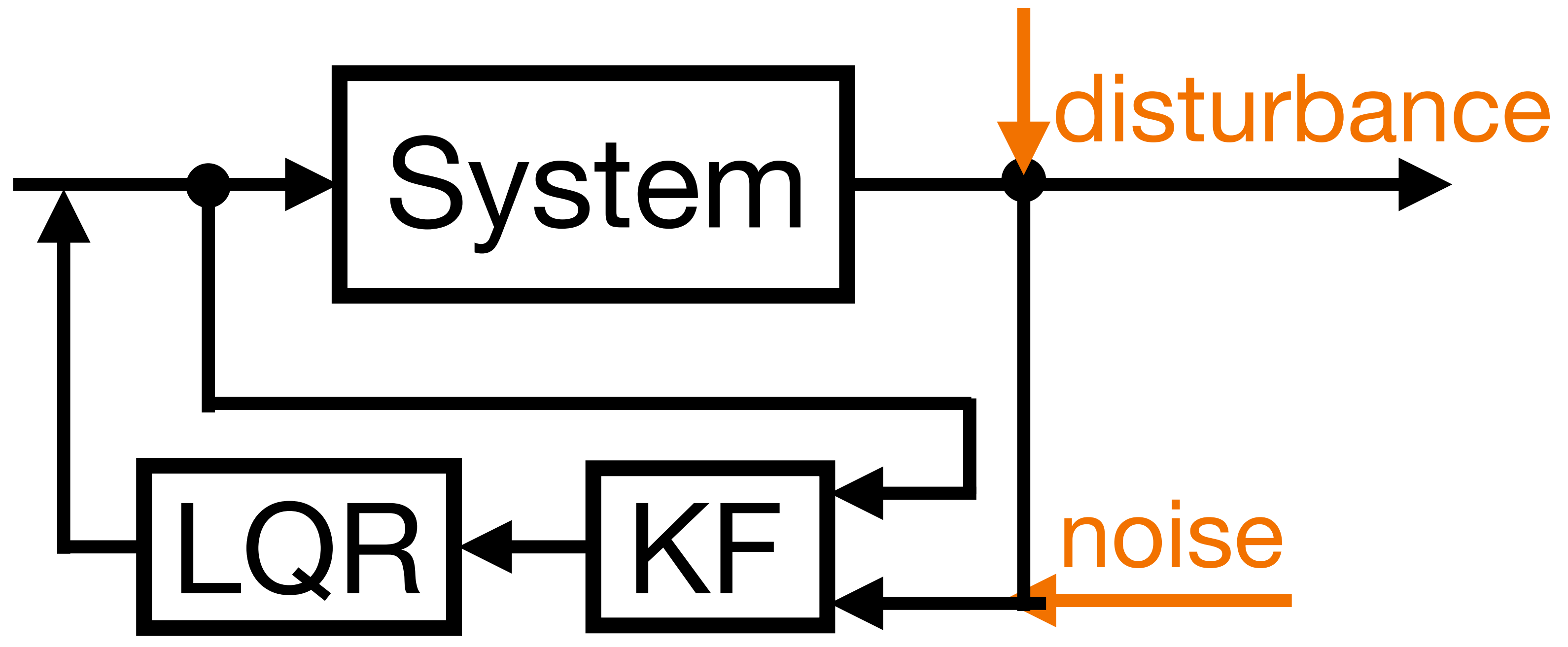
prediction

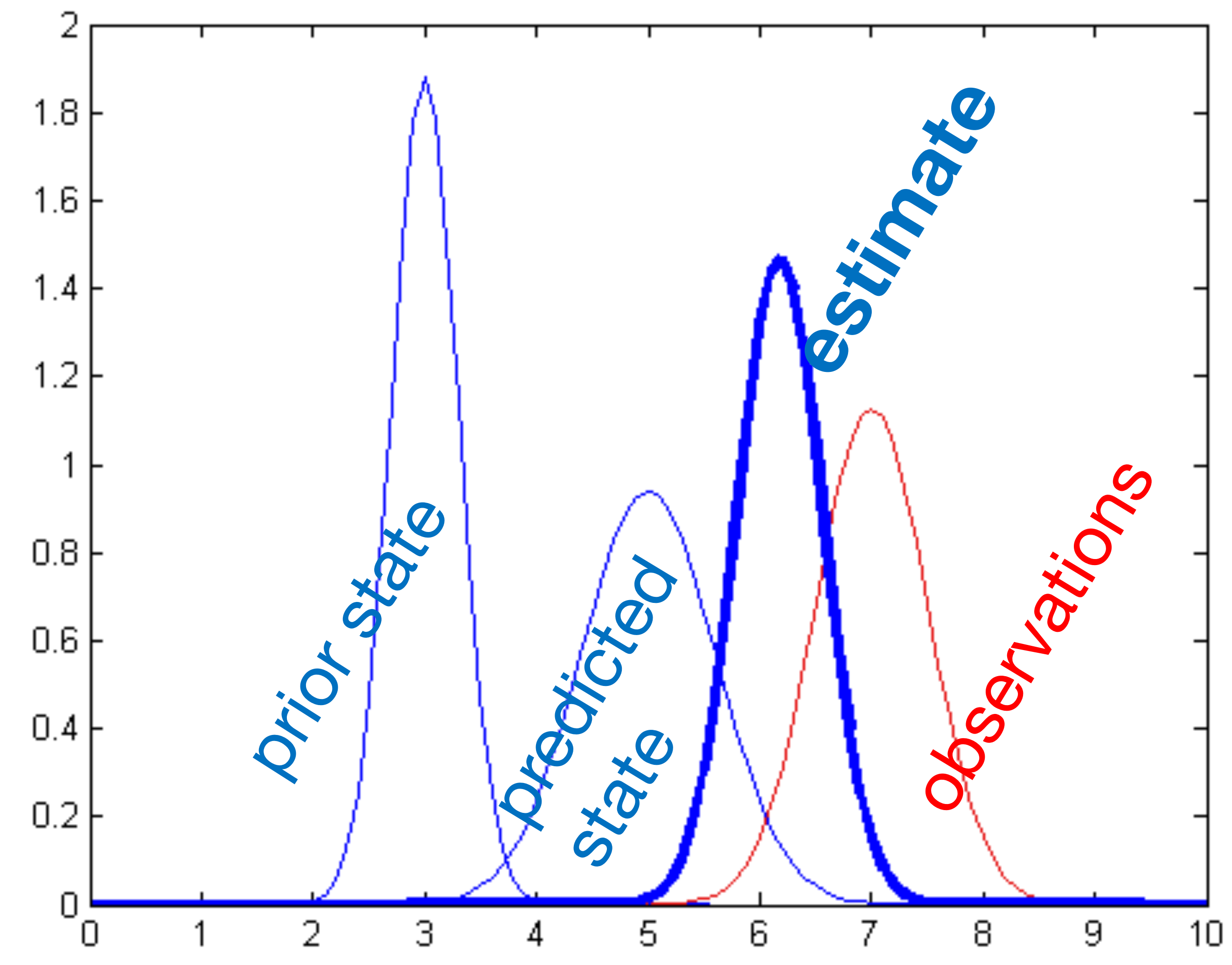
1. $\mu_p(t) = A\mu(t-1) + Bu(t)$
2. $\Sigma_p(t) = A\Sigma(t-1)A^T + \Sigma_u$

update

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 $\Sigma(t)$

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. $\mu_p(t) = A\mu(t-1) + Bu(t)$
2. $\Sigma_p(t) = A\Sigma(t-1)A^T + \Sigma_u$

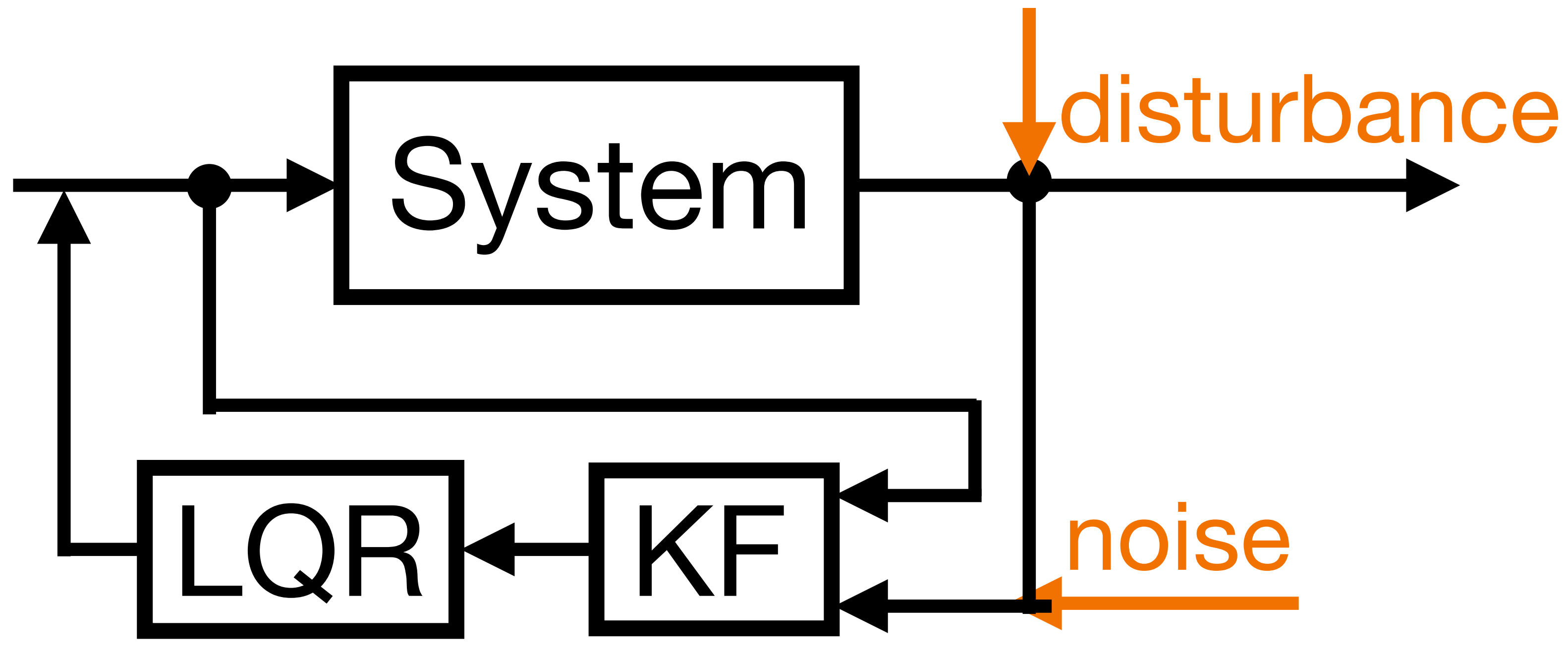
prediction

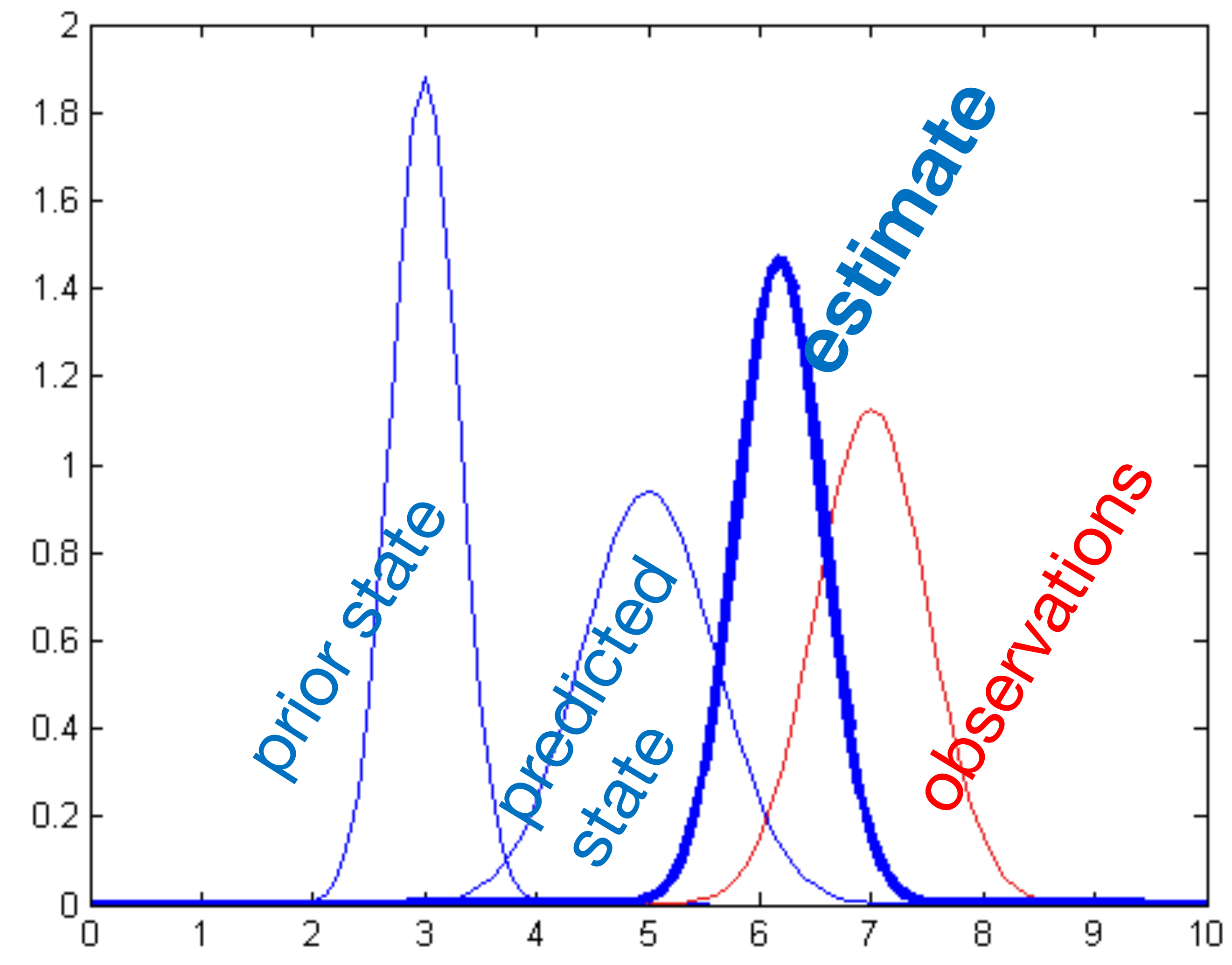
3. $K_{KF} = \Sigma_p(t)C^T(C\Sigma_p(t)C^T + \Sigma_z)^{-1}$
4. $\mu(t) = \mu_p(t) + K_{KF}(z(t) - C\mu_p(t))$
5. $\Sigma(t) = (I - K_{KF}C)\Sigma_p(t)$

update

6. Return $\mu(t)$ and $\Sigma(t)$

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

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

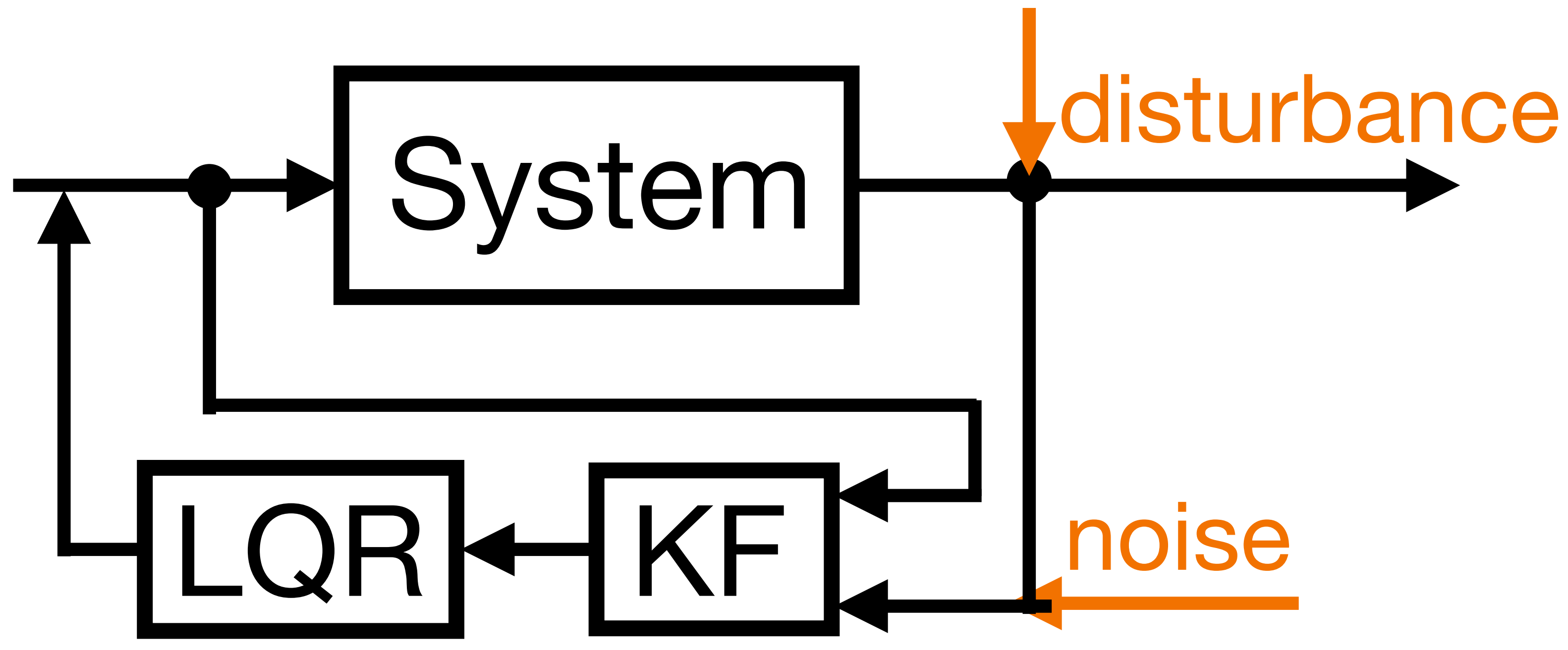
Kalman Filter $(\mu(t-1), \Sigma(t-1), u(t), z(t))$

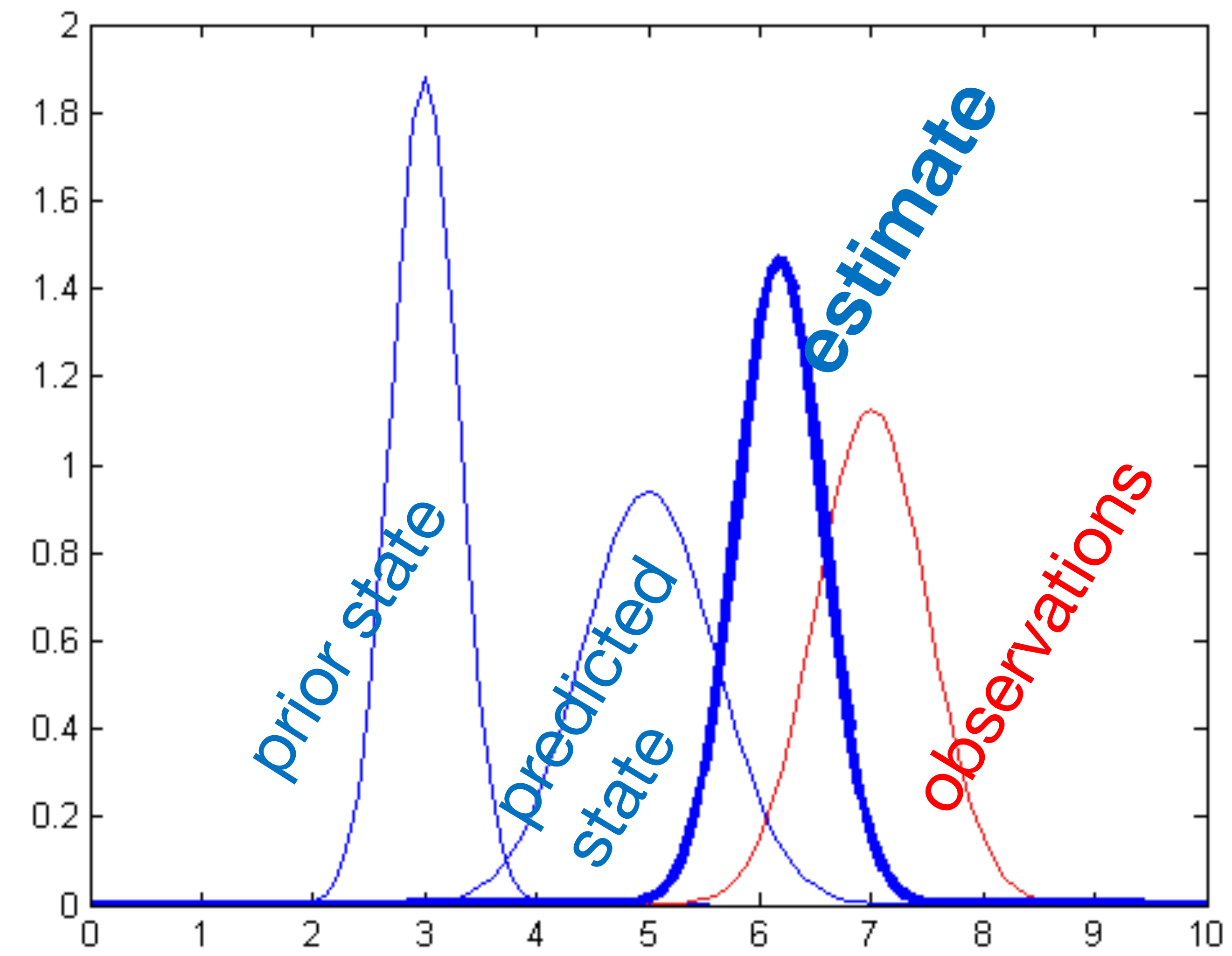
prediction

1. $\mu_p(t) = A\mu(t-1) + Bu(t)$
2. $\Sigma_p(t) = A\Sigma(t-1)A^T + \Sigma_u$

update

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 $\Sigma(t)$





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

```
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))
3.    bel(x(t)) = α p(z(t)|x(t))bel(x(t))
4. end for
5. return bel(x_t)
```

Lab 5-8: PID control — Sensor Fusion — Stunt

- Labs 5 and 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)
  - Perform a step response with the robot and build the state space equations
  - Estimate covariance matrices for process and sensor noise
  - Try the Kalman Filter in Jupyter with your own data from Lab 5
  - Implement the Kalman Filter on your robot
- Lab 8: Use KF + PID to execute fast stunts

Lab 7: Kalman Filter

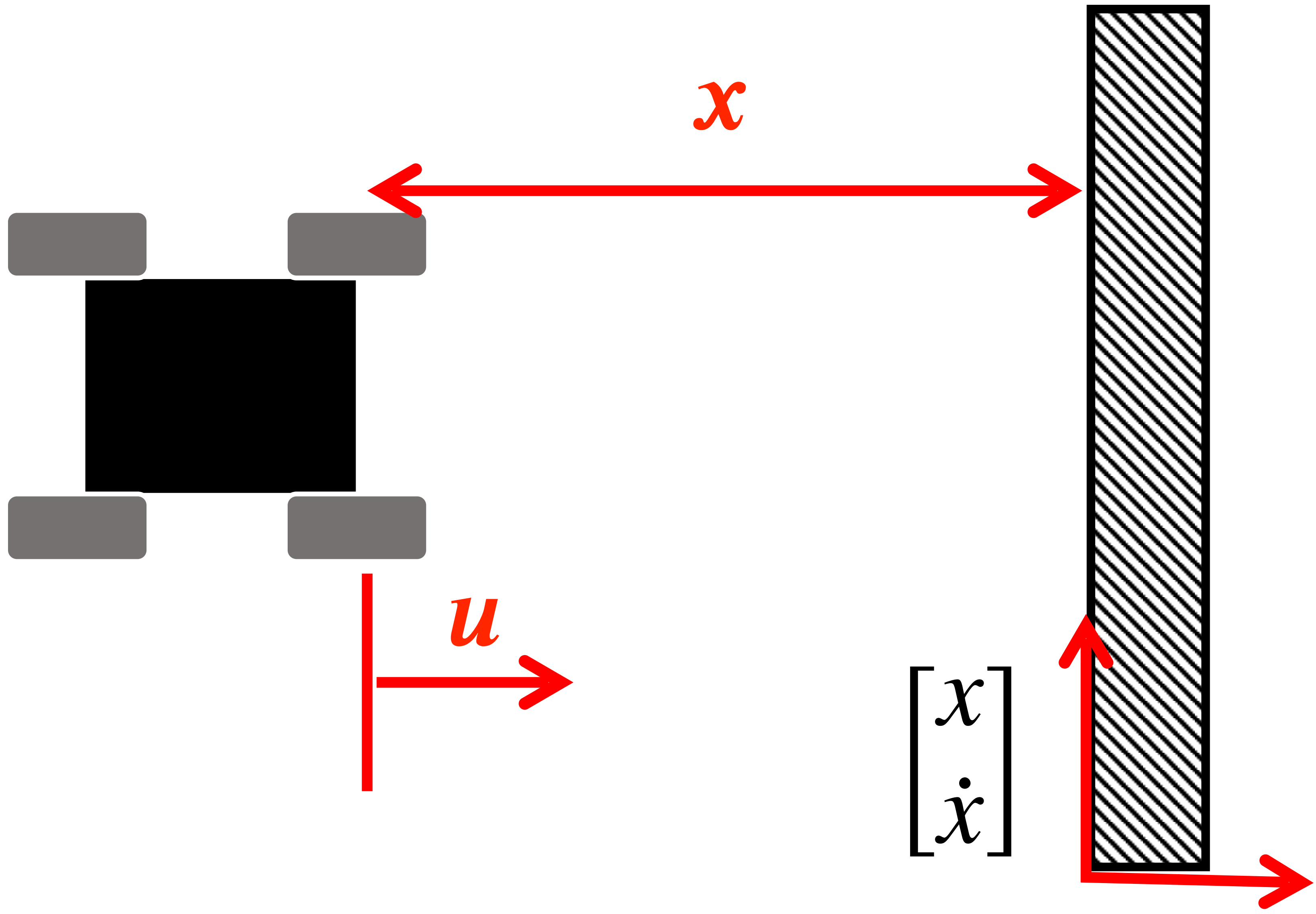
$$F = ma = m\ddot{x}$$

$$F = u - \dot{x}$$

$$m\ddot{x} = u - \dot{x}$$

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

What are d and m?



State space equations

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

$$C = [-1 \quad 0]$$

Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$

$$F = u - \dot{x}$$

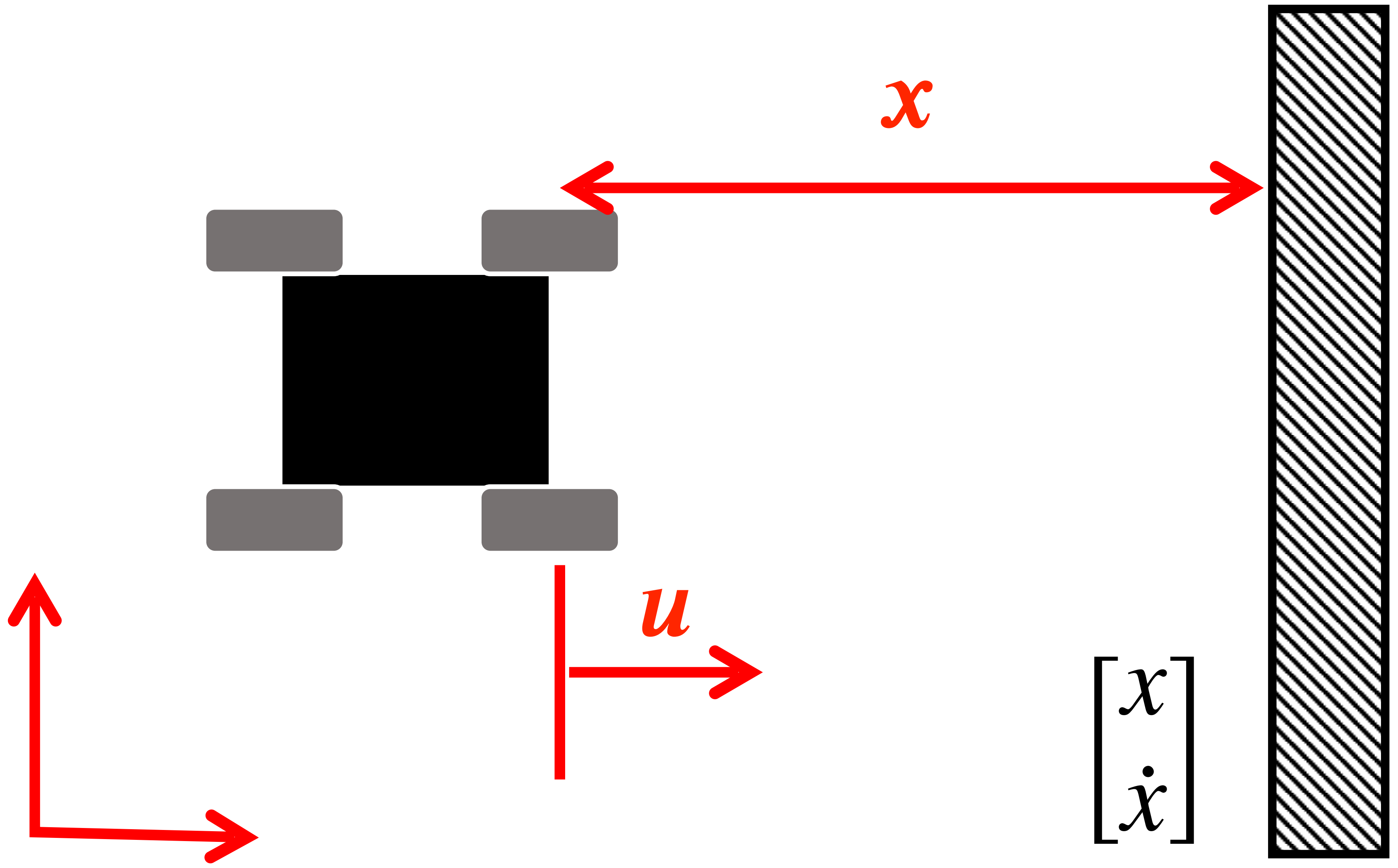
$$m\ddot{x} = u - \dot{x}$$

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

What are d and m?

At constant speed, we can find d:

$$0 = \frac{u}{m} - \frac{d}{m}\dot{x} \qquad d = \frac{u}{\dot{x}}$$



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

State space equations

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

$$C = [-1 \quad 0]$$

Lab 7: Kalman Filter

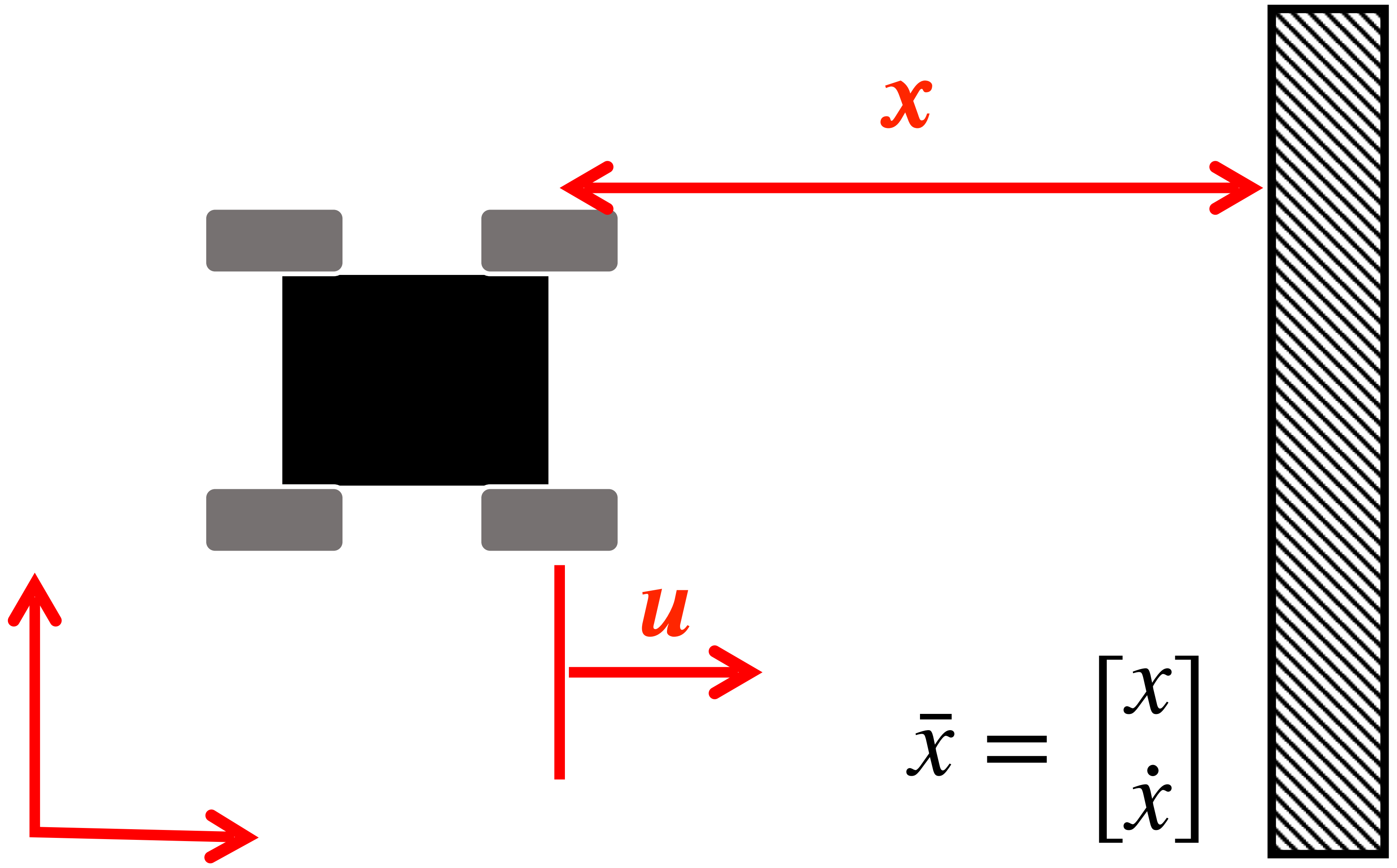
$$F = ma = m\ddot{x}$$

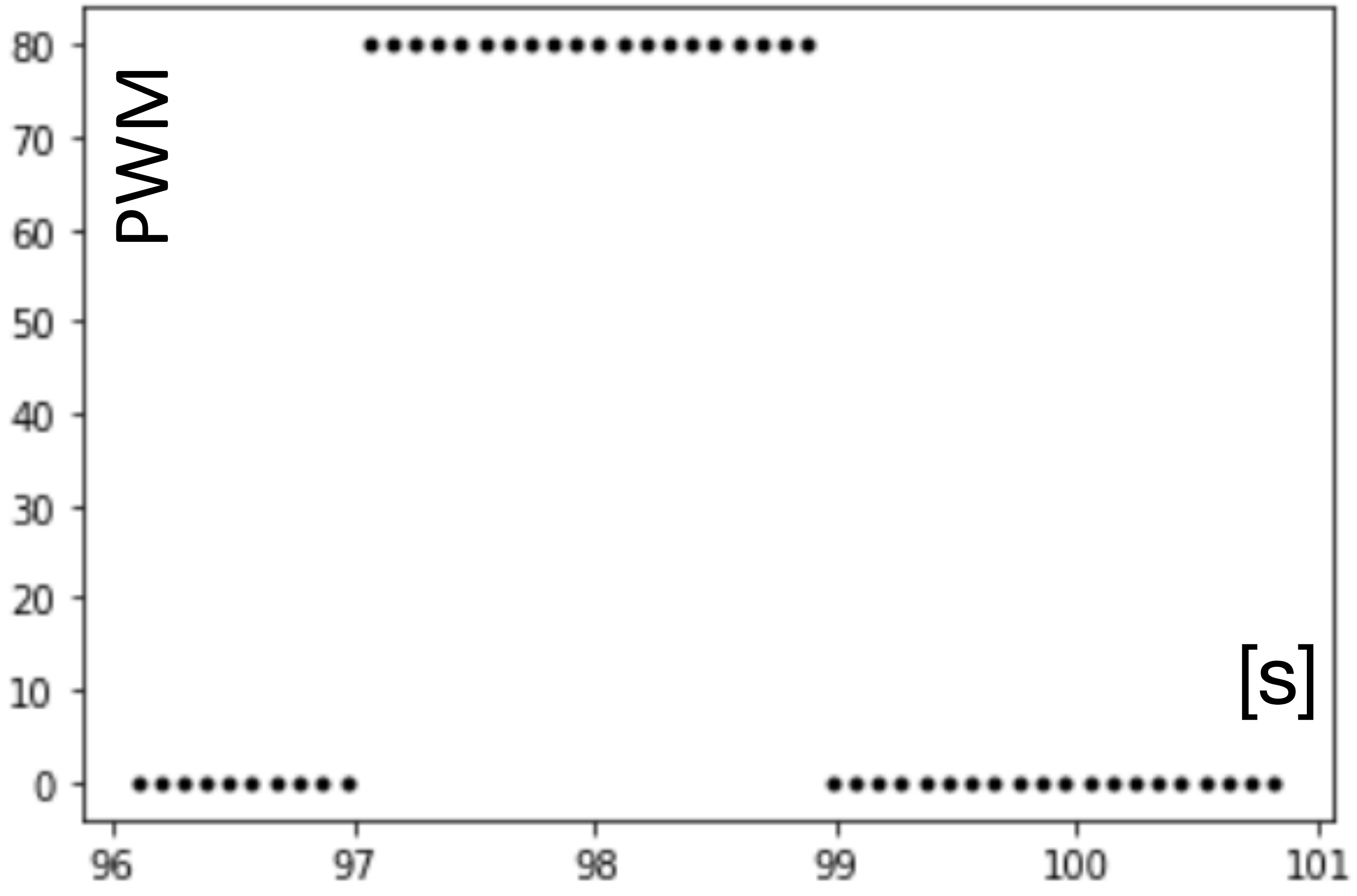
$$F = u - \dot{x}$$

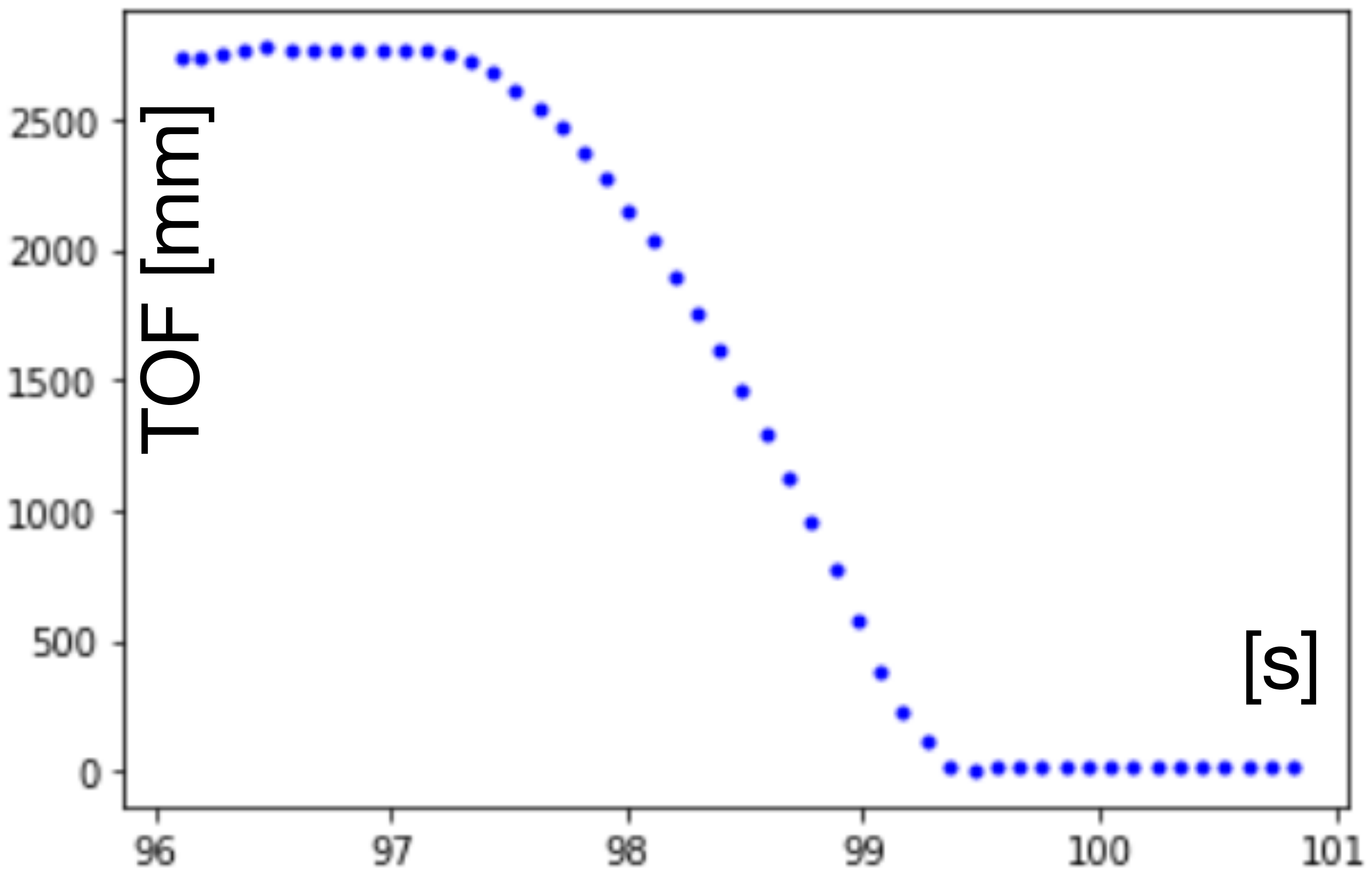
$$m\ddot{x} = u - \dot{x}$$

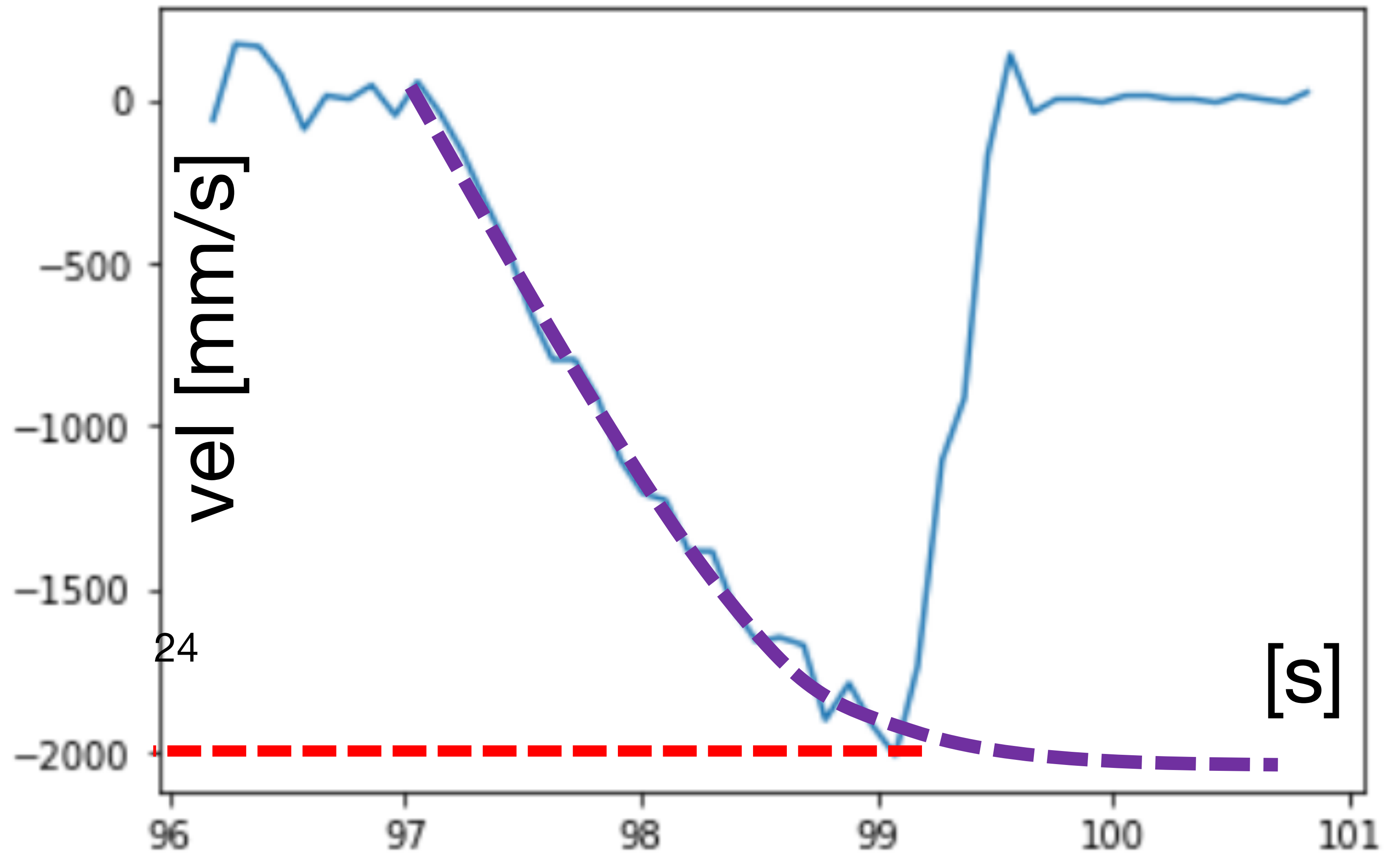
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

What are d and m?









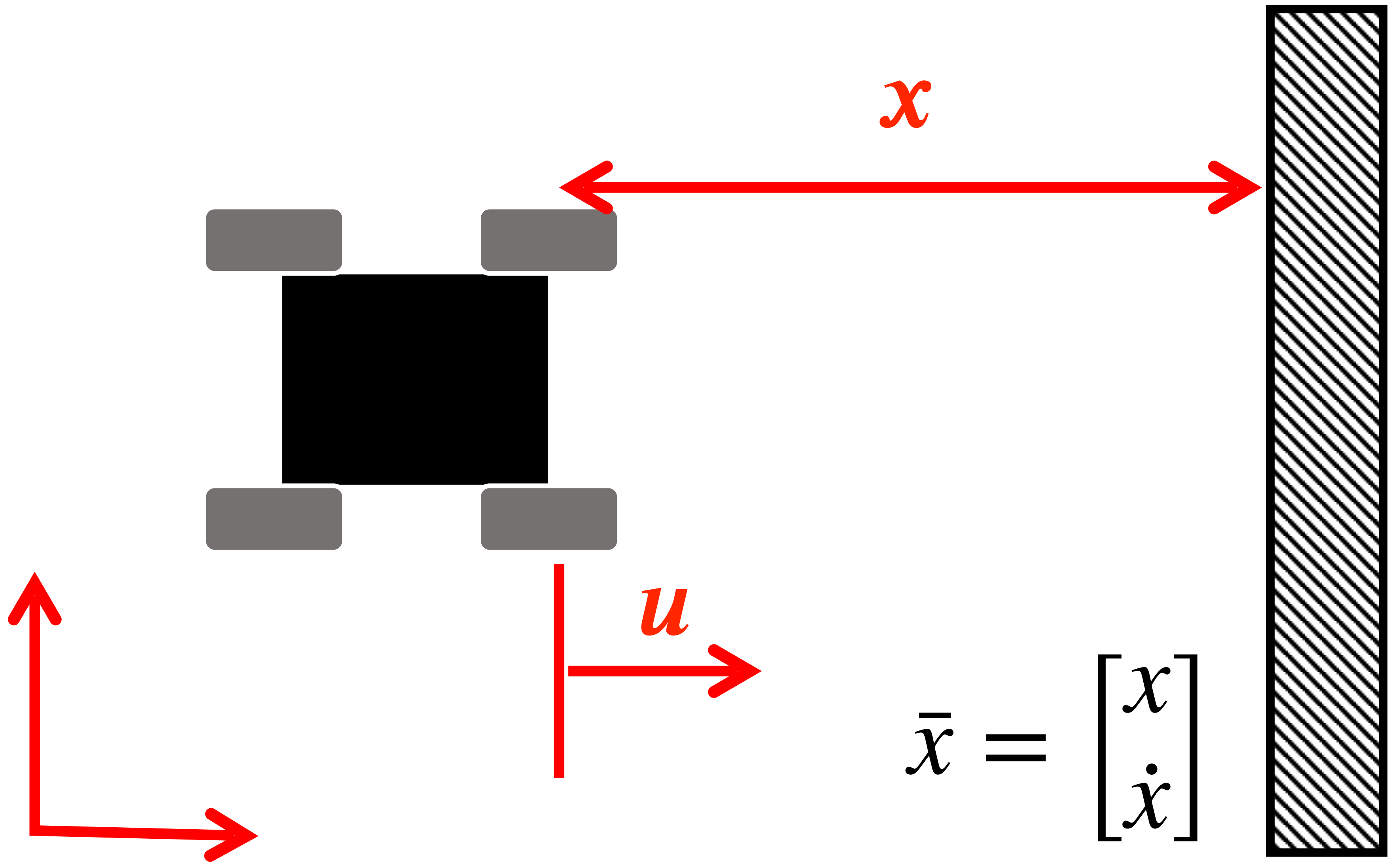
Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$

$$F = u - \dot{x}$$

$$m\ddot{x} = u - \dot{x}$$

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



What are d and m?

At constant speed, we can find d:

$$0 = \frac{u}{m} - \frac{d}{m}\dot{x} \qquad d = \frac{u}{\dot{x}}$$

(assume u=1 for now)

$$d \approx \frac{1}{2000\text{mm/s}}$$

State space equations

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

$$C = [-1 \quad 0]$$

Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$

$$F = u - \dot{x}$$

$$m\ddot{x} = u - \dot{x}$$

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

1st order system:

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

Unit step response solution:

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

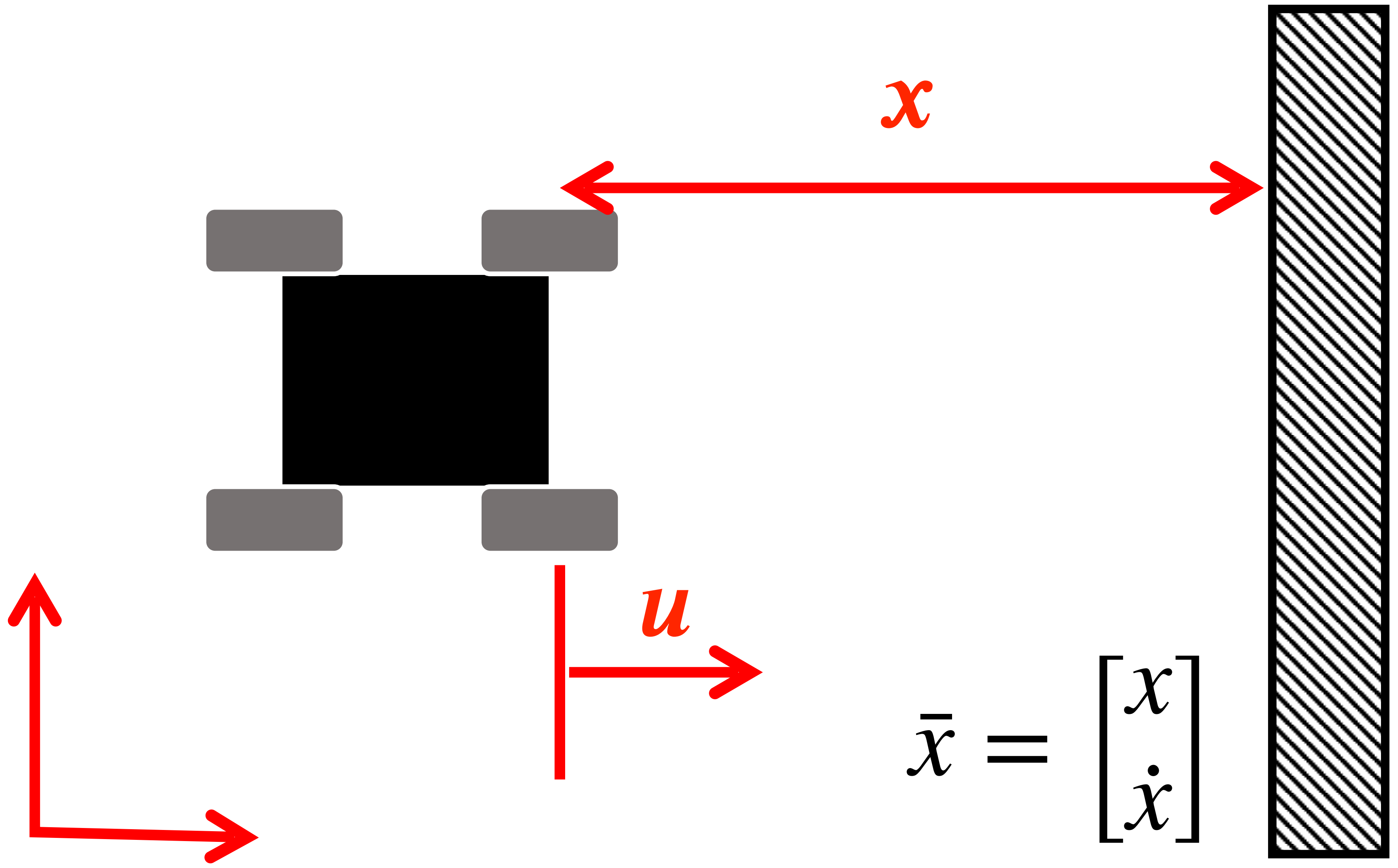
What are d and m?

Use the rise time to determine m

$$\dot{\nu} = \frac{u}{m} - \frac{d}{m}\nu$$

$$\nu = 1 - e^{-\frac{d}{m}t_{0.9}} \qquad \ln(1-\nu) = -\frac{d}{m}t_{0.9}$$

$$m = \frac{-dt_{0.9}}{\ln(1-0.9)}$$



State space equations

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

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

Lab 7: Kalman Filter

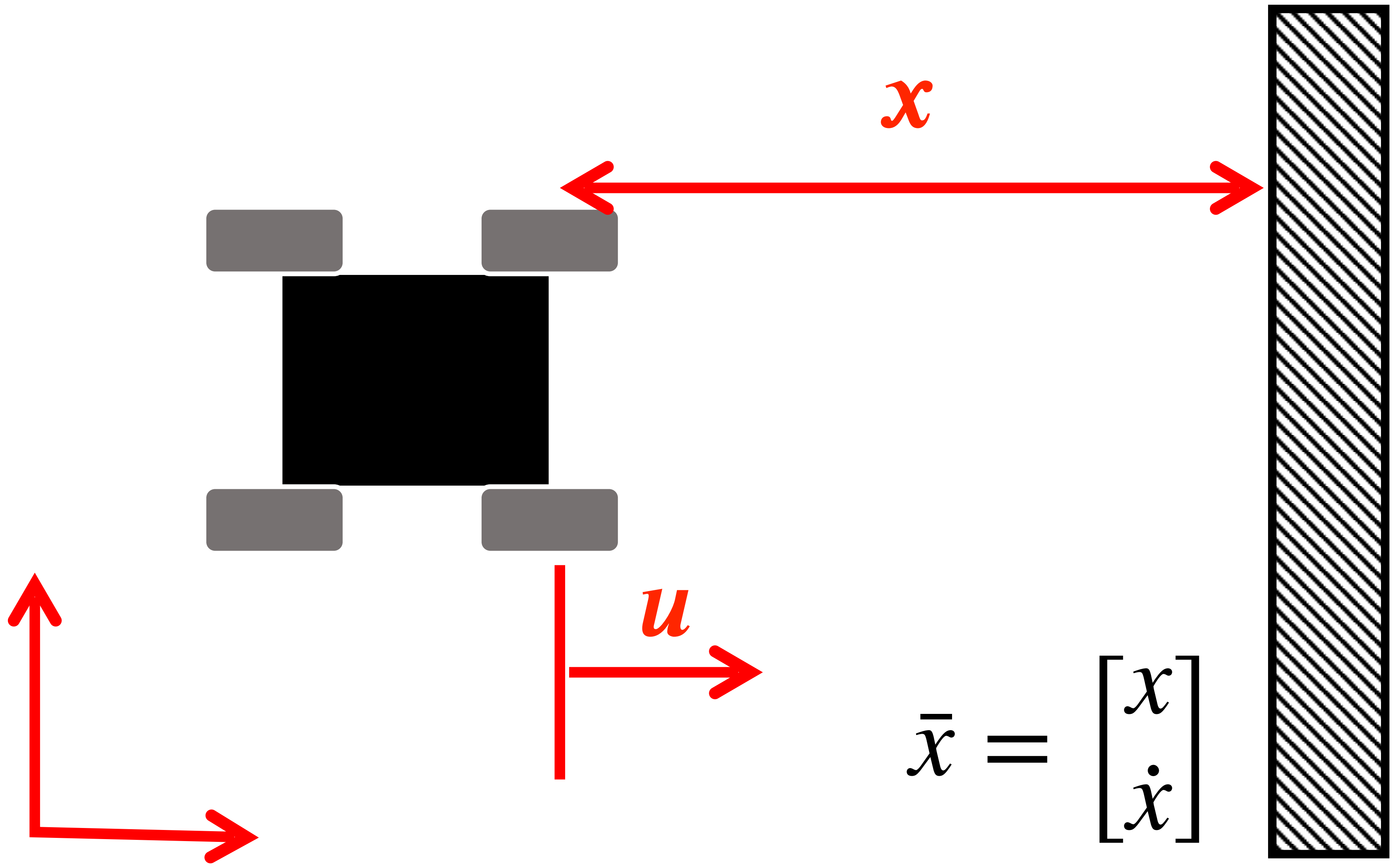
$$F = ma = m\ddot{x}$$
$$F = u - \dot{x}$$
$$m\ddot{x} = u - \dot{x}$$
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

1st order system:

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

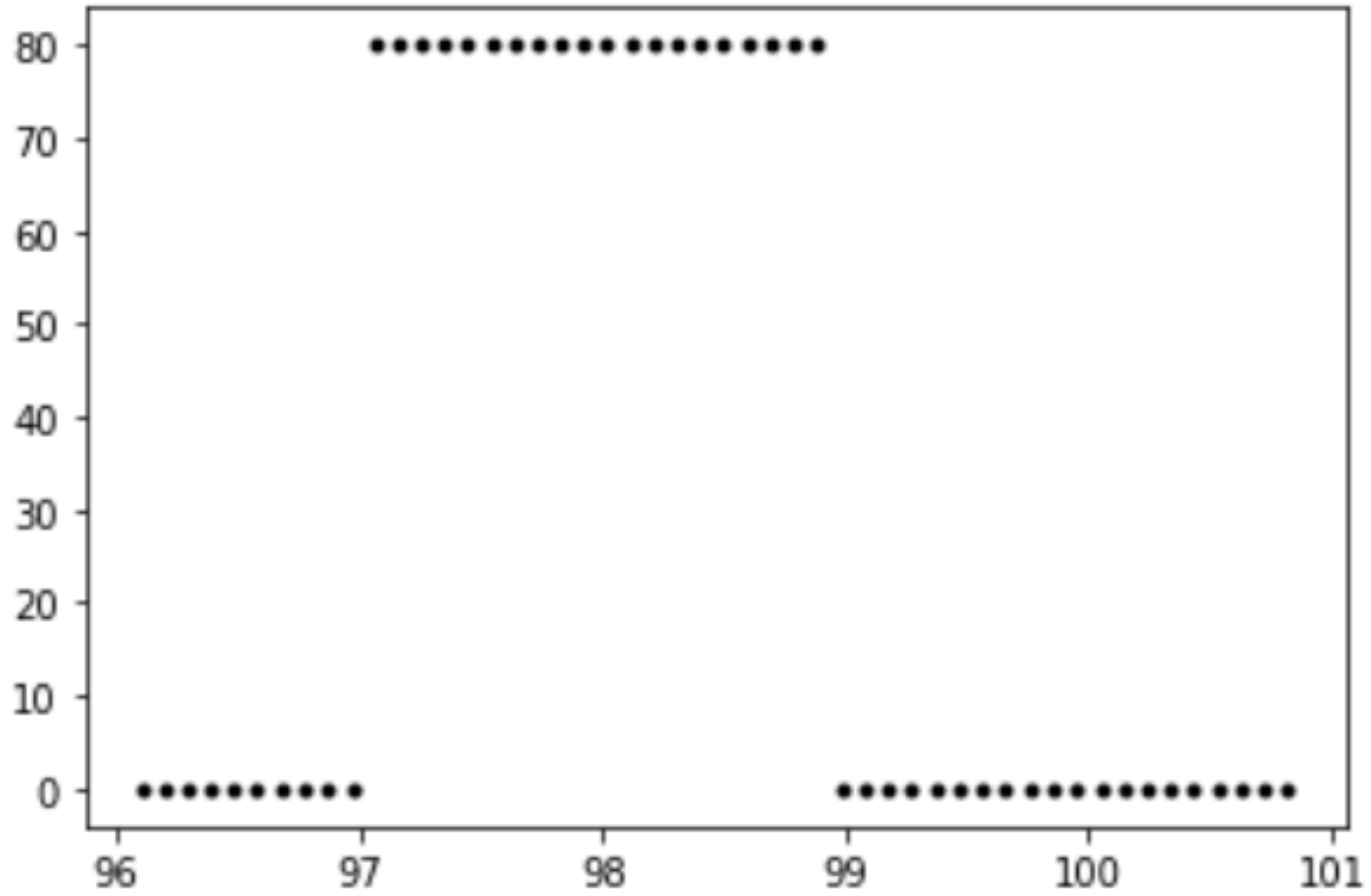
Unit step response solution:

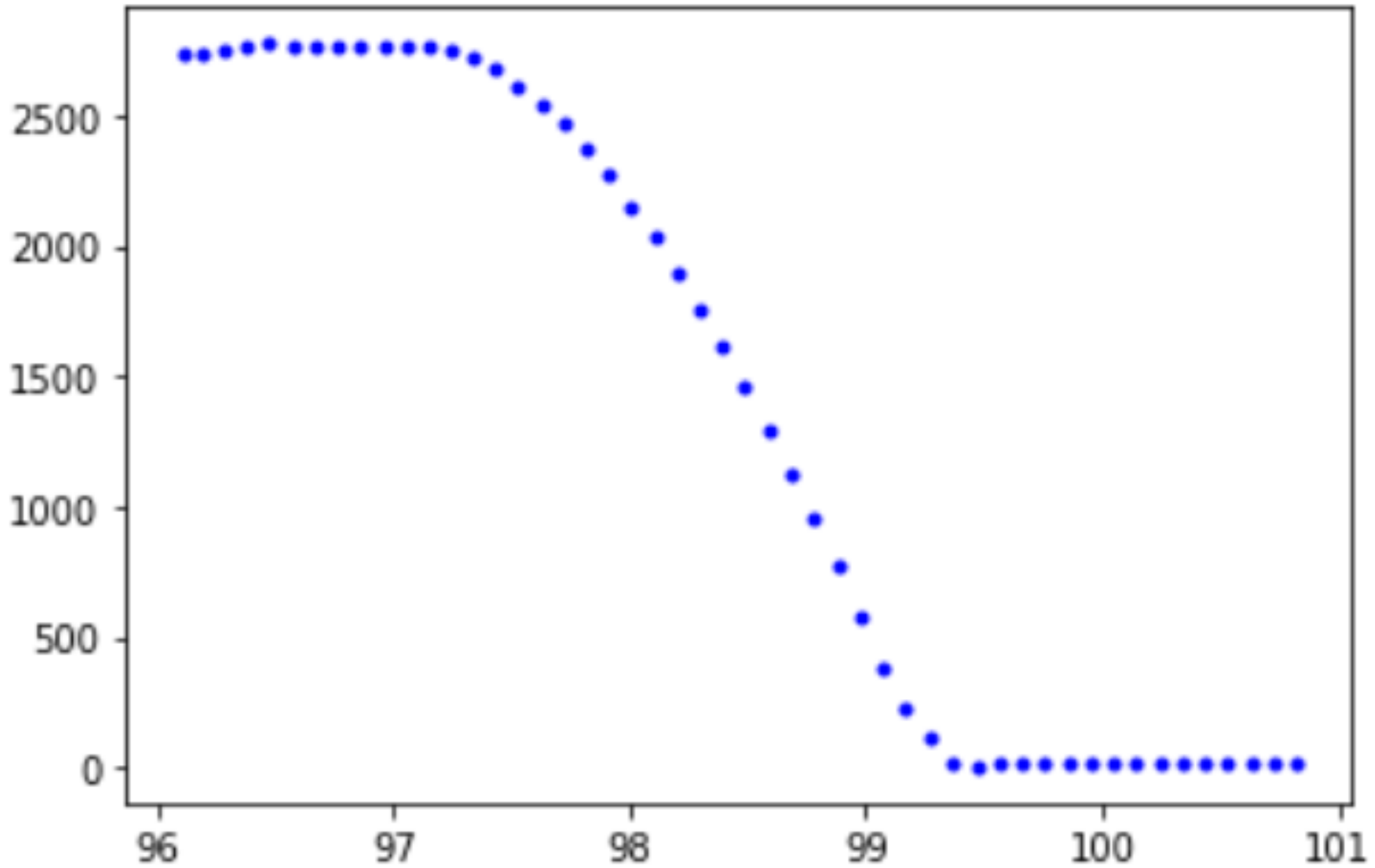
$$y(t) = 1 - e^{-\frac{t}{\tau}}$$

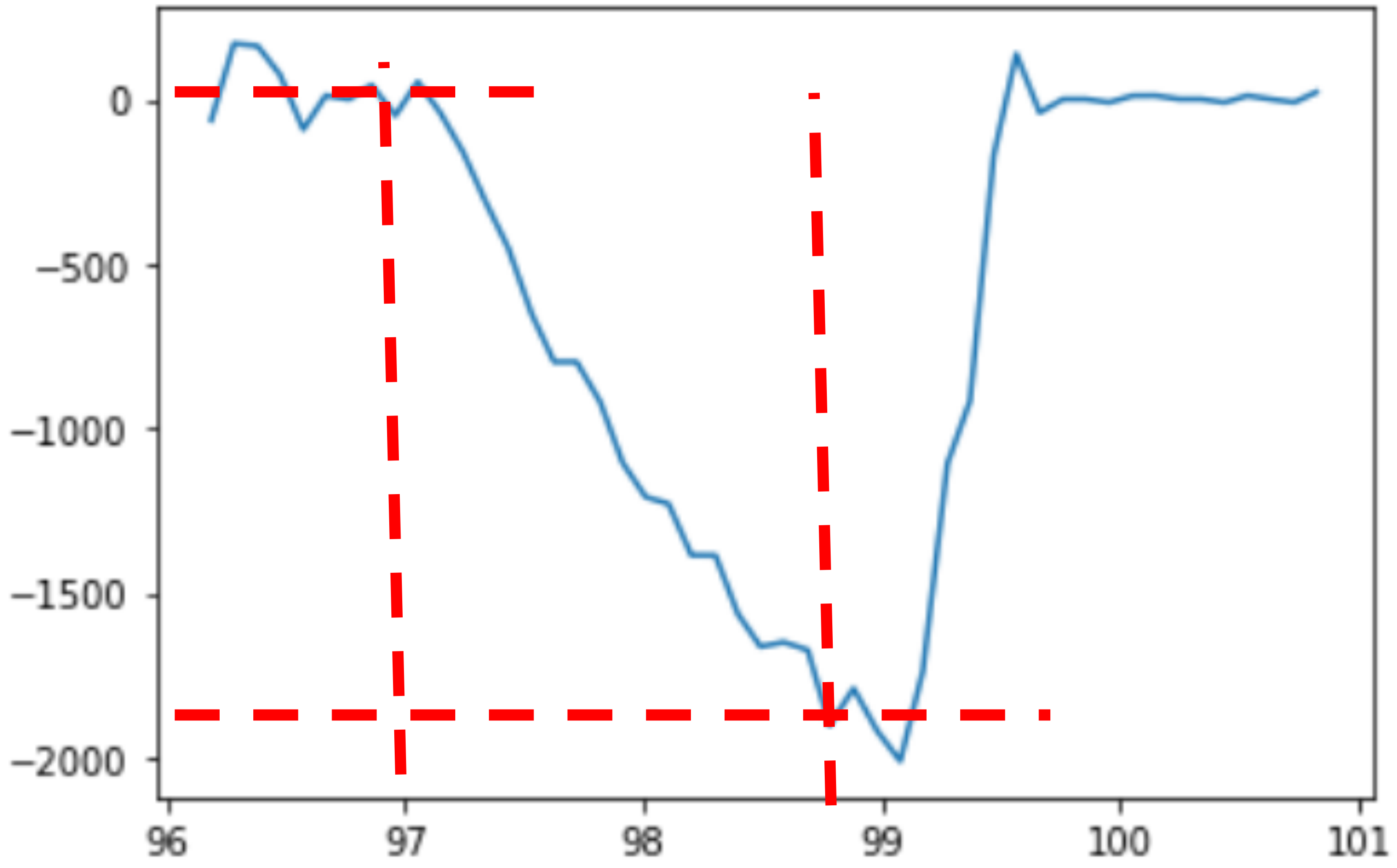


What are d and m?

Use the rise time to determine m







Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$

$$F = u - \dot{x}$$

$$m\ddot{x} = u - \dot{x}$$

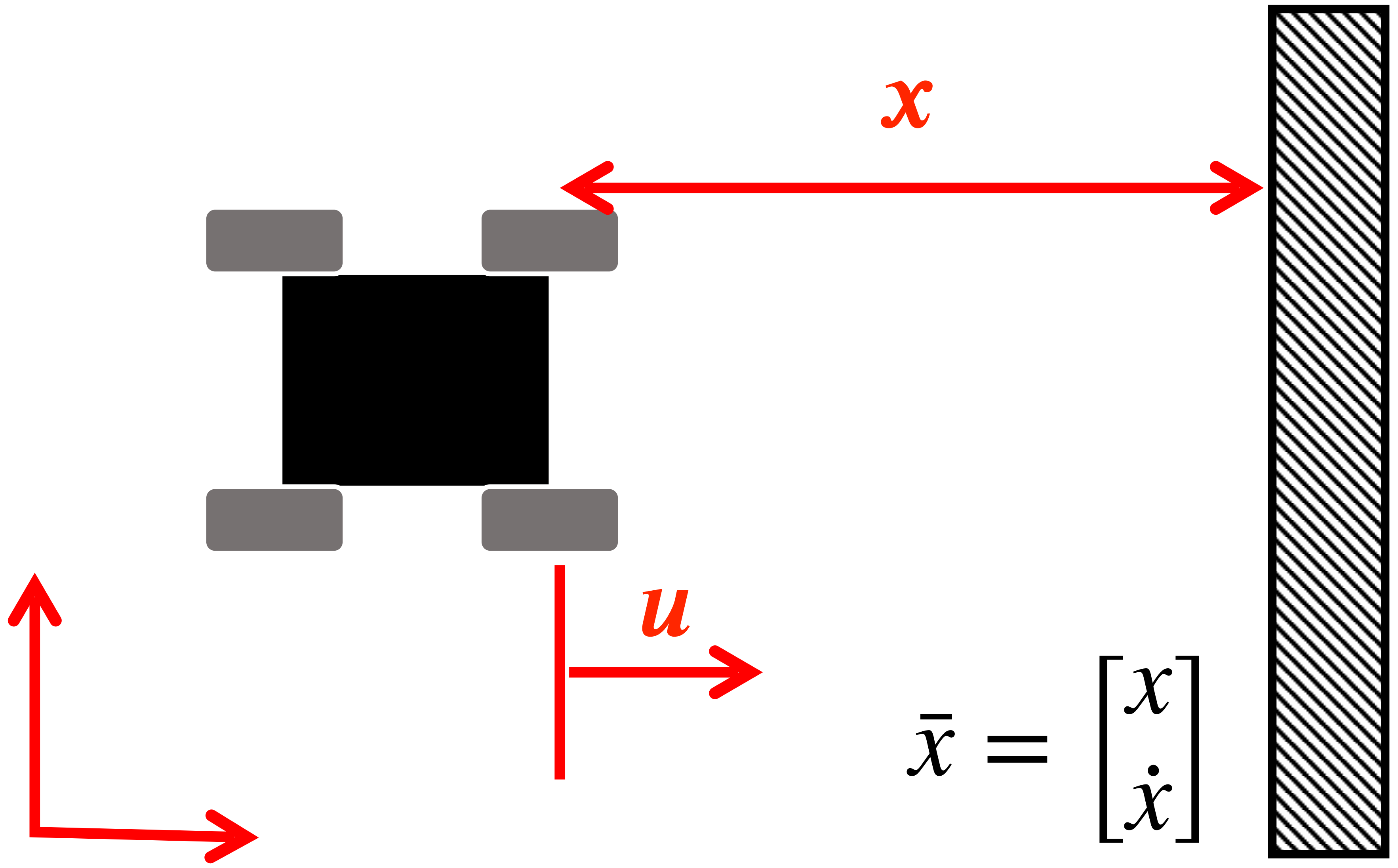
$$\ddot{x} = \frac{u}{m} - \frac{d}{m}\dot{x}$$

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 are d and m?

Use the rise time to determine m

$$\dot{v} = \frac{u}{m} - \frac{d}{m}v$$

$$v = 1 - e^{-\frac{d}{m}t_{0.9}} \qquad \ln(1 - v) = -\frac{d}{m}t_{0.9}$$

$$m = \frac{-dt_{0.9}}{\ln(1-0.9)} = \frac{-0.0005 \cdot 1.9}{\ln(0.1)} = 4.1258 \cdot 10^{-4}$$

State space equations

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

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

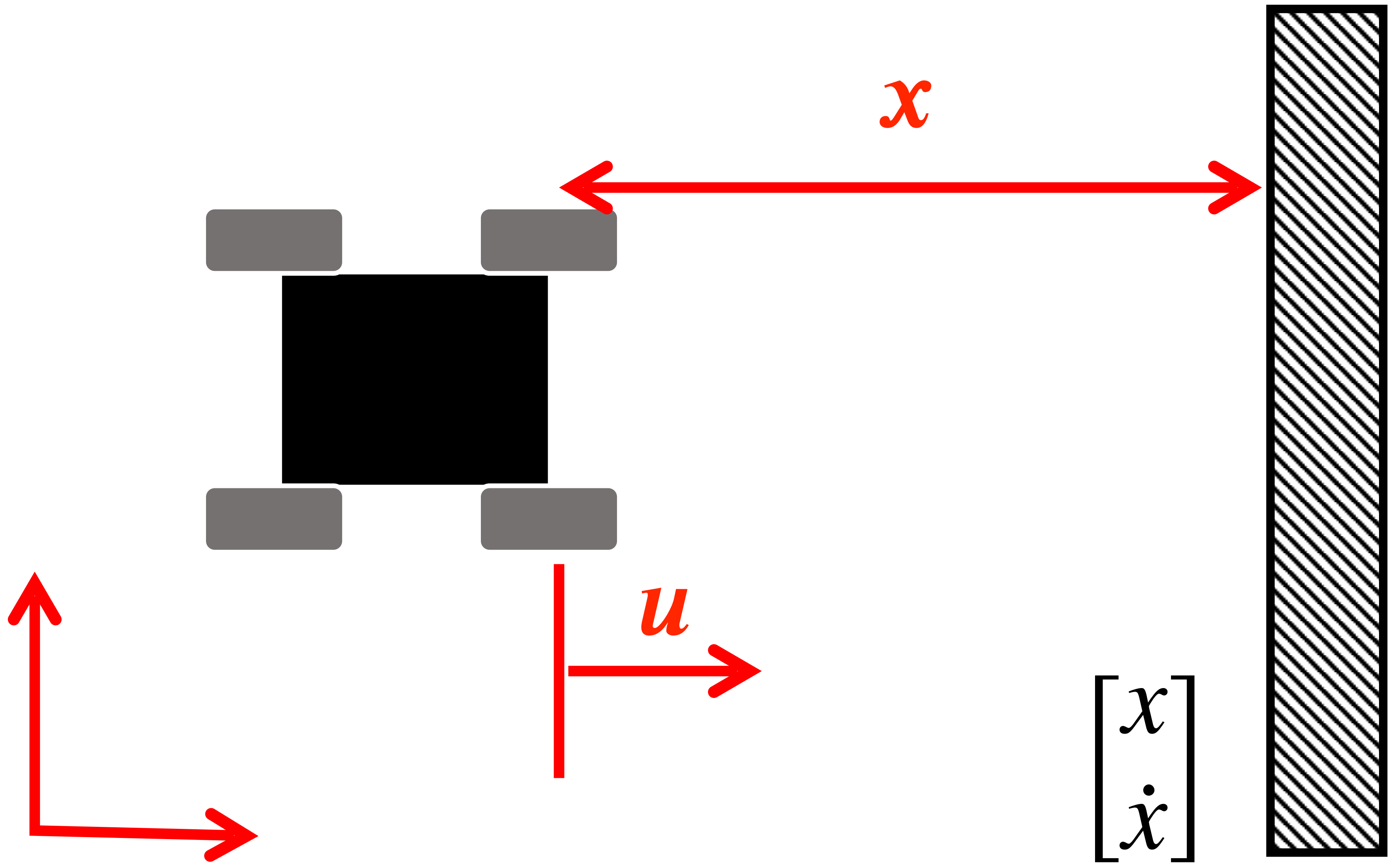
Lab 7: Kalman Filter

$$F = ma = m\ddot{x}$$

$$F = u - \dot{x}$$

$$m\ddot{x} = u - \dot{x}$$

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



What are d and m?

At steady state (constant 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

$$m = \frac{-dt_{0.9}}{\ln(1-0.9)} \approx 4.1258 \cdot 10^{-4}$$

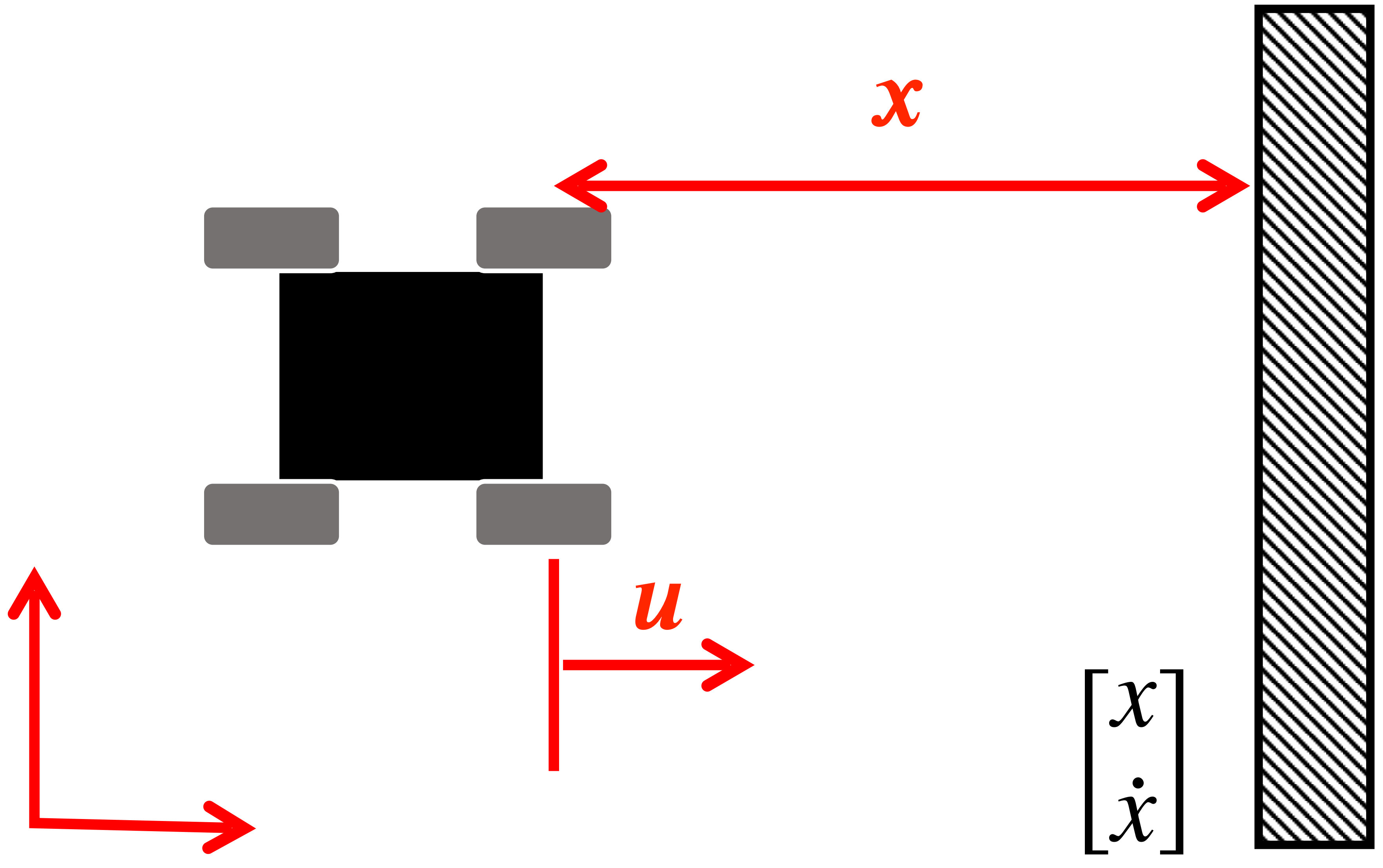
State space equations

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

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

Lab 7: Kalman Filter

- We have $A,\ B,\ C,\ \Sigma_u,\ \Sigma_z$
- Discretize the A and B matrices
  - $x(n+1) = x(n) + dx$
  - $dx/dt = Ax + Bu \iff dx = dt(Ax + Bu)$
  - $x(n+1) = x(n) + dt(Ax(n) + Bu)$
  - $x(n+1) = \underbrace{(I + dt \cdot A)}_{A_d}x(n) + \underbrace{dt \cdot B}_{B_d}u$
  - dt is our sampling time (0.130s)
- Rescale from unity input to actual input



State space equations

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

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

Lab 7: Kalman Filter

Implement the Kalman Filter

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

1. $\mu_p(t) = A\mu(t-1) + Bu(t)$
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 $\Sigma(t)$

Next, determine measurement and process noise

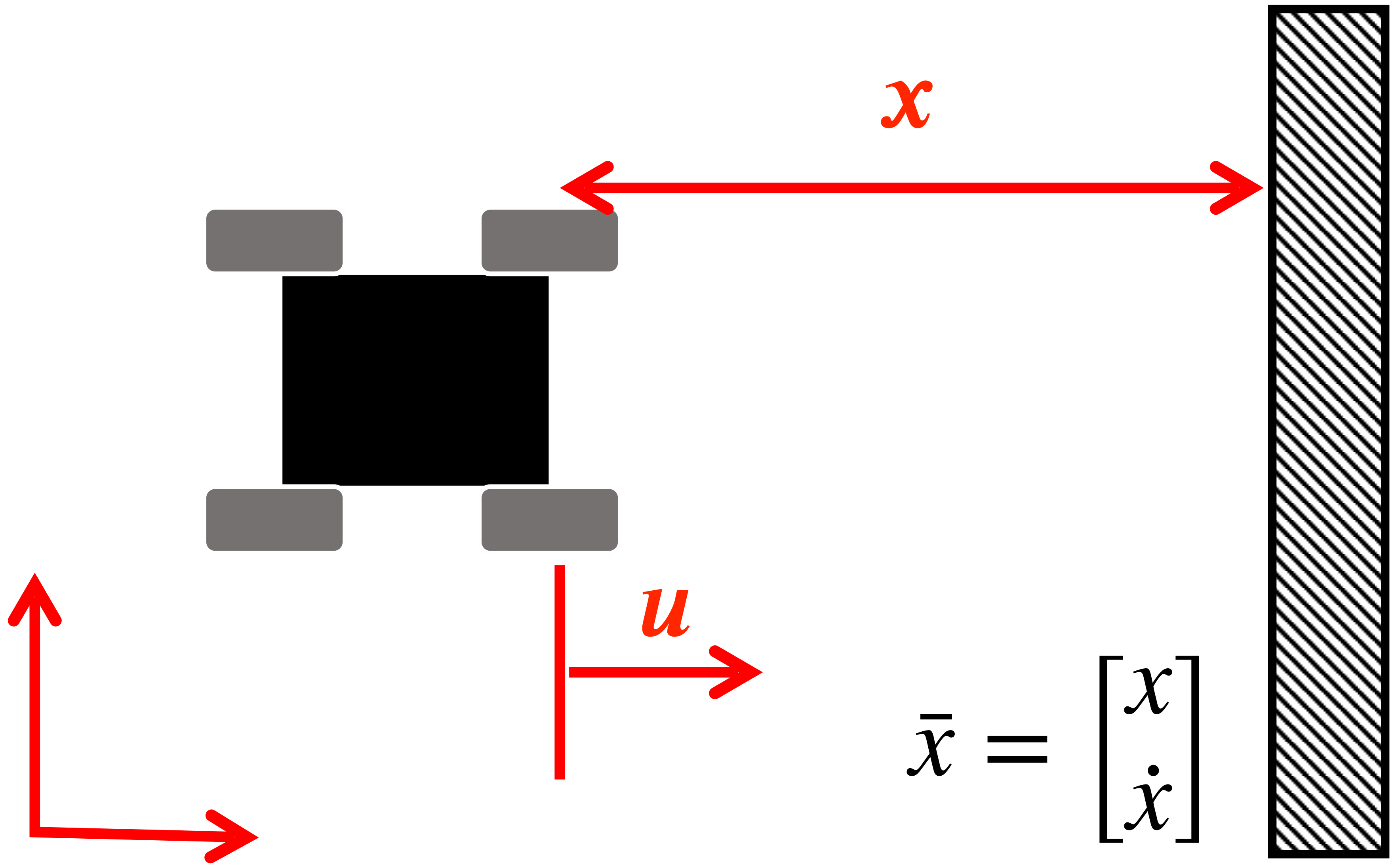
```
def kf(mu,sigma,u,y):

    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
```

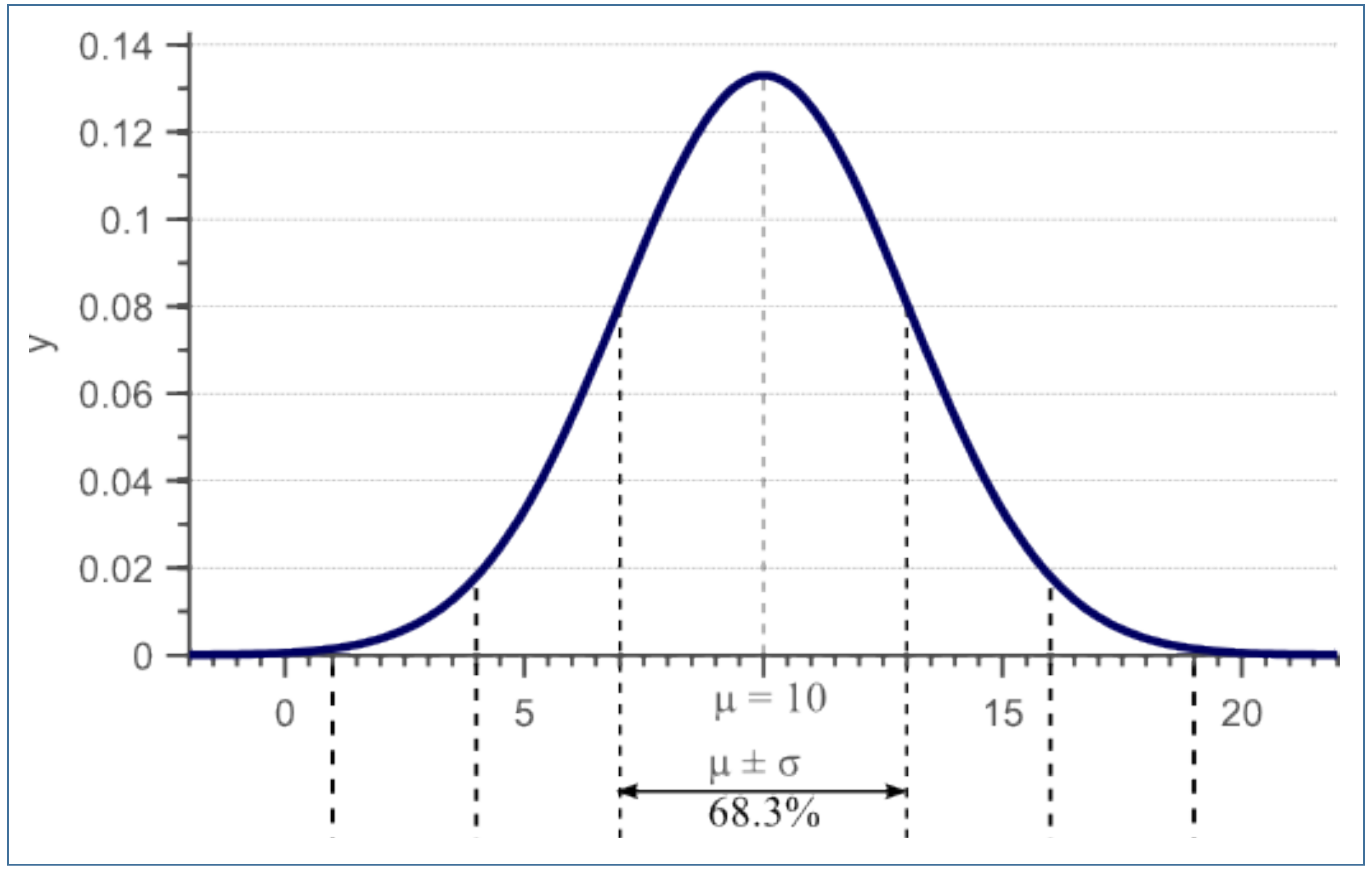
Lab 7: Kalman Filter

Implement the Kalman Filter

- Measurement noise
  - $\Sigma_z = [\sigma_3^2]$
  - $\sigma_3^2 = (20\text{mm})^2$
- 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$_{\text{stddev}}$ after 1s: $\sqrt{10^2 \cdot \frac{1}{0.13}} = 27.7\text{mm}$
  - Trust in modeled speed:
    - Speed$_{\text{stddev}}$ after 1s: $\sqrt{10^2 \cdot \frac{1}{0.13}} = 27.7\text{mm/s}$

Sample time ~0.13s





Lab 7: Kalman Filter

Implement the Kalman Filter

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

1. $\mu_p(t) = A\mu(t-1) + Bu(t)$
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 $\Sigma(t)$

Finally, determine your initial state mean and covariance

$$\mu(t-1)$$

$$\Sigma(t-1)$$

```
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