Georgia Institute of Technology

Dr. Magnus Egerstedt
Professor
School of Electrical and
Computer Engineering

Control of Mobile Robots

Module 4
Control Design

How make mobile robots move in effective, safe, predictable, and collaborative ways using modern control theory?

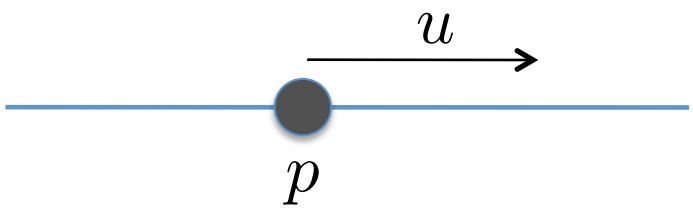
Lecture 4.1 – Stabilizing the Point Mass

- Given a linear system

$$\dot{x} = Ax + Bu, \quad y = Cx$$

- The dilemma: We seem to need x but all we have is y
- Game plan:
  - Design u as if we had x
  - Figure out x from y
- Step 1: State feedback using “pole placement”

Back to the Point Mass



$$\dot{x} = Ax + Bu, \quad A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- State Feedback: $u = -Kx = -\begin{bmatrix} k_1 & k_2 \end{bmatrix} x$
- How pick the control gains?
- Idea: Pick them in order to make the closed-loop system have the desired eigenvalues (poles)

Computing Eigenvalues

- Given a matrix M, its eigenvalues satisfy the characteristic equation

$$\chi_M(\lambda) = \det(\lambda I - M) = 0$$

$$M = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix}$$

$$\lambda I - M = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - M = \begin{bmatrix} \lambda - m_1 & -m_2 \\ -m_3 & \lambda - m_4 \end{bmatrix}$$

Computing Eigenvalues

- Given a matrix M, its eigenvalues satisfy the characteristic equation

$$\chi_M(\lambda) = \det(\lambda I - M) = 0$$

$$\det(\lambda I - M) = \begin{vmatrix} \lambda - m_1 & -m_2 \\ -m_3 & \lambda - m_4 \end{vmatrix}$$

$$= (\lambda - m_1)(\lambda - m_4) - m_2 m_3$$

$$= \lambda^2 - (m_1 + m_4)\lambda + m_1 m_4 - m_2 m_3 = 0$$

Computing Eigenvalues

- Given a matrix M, its eigenvalues satisfy the characteristic equation

$$\chi_M(\lambda) = \det(\lambda I - M) = 0$$

$$\chi_M(\lambda) = \lambda^2 - (m_1 + m_2)\lambda + m_1 m_4 - m_2 m_3 = 0$$

$$\lambda = \frac{m_1 + m_4}{2} \pm \sqrt{\frac{(m_1 + m_4)^2}{4} - m_1 m_4 + m_2 m_3}$$

- But this is really annoying – is there an easier way?

Almost Computing Eigenvalues

- The fundamental theorem of algebra: The roots in a polynomial are completely determined by the coefficients

$$\chi_M(\lambda) = \lambda^2 - (m_1 + m_2)\lambda + m_1 m_4 - m_2 m_3 = 0$$

- Let's stop here!

Back to the Point-Mass, Again

$$u = -Kx \;\Rightarrow\; \dot{x} = (A - BK)x$$

$$A - BK = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix}$$

$$\chi_{A-BK}(\lambda) = \begin{vmatrix} \lambda & -1 \\ k_1 & \lambda + k_2 \end{vmatrix} = \lambda^2 + \lambda k_2 + k_1$$

Desired Eigenvalues

- Let's pick the eigenvalues that we would like the closed-loop system to have

$$\lambda_1, \ldots, \lambda_n$$

- If these were indeed the eigenvalues, then the characteristic polynomial would be

$$\varphi(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n) = \prod_{i=1}^{n} (\lambda - \lambda_i)$$

- For the robot, let's pick both eigenvalues at -1

$$\varphi(\lambda) = (\lambda + 1)(\lambda + 1) = \lambda^2 + 2\lambda + 1$$

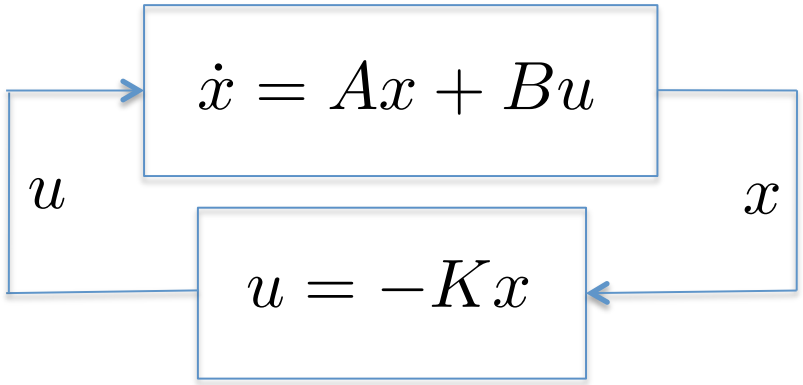
- Now we just line up the coefficients!

Lining Up the Coefficients

$$\chi_{A-BK} = \lambda^2 + k_2\lambda + k_1$$

$$\varphi(\lambda) = (\lambda + 1)(\lambda + 1) = \lambda^2 + 2\lambda + 1$$

$$\begin{array}{ll} \lambda^1: & k_2 = 2 \\ \lambda^0: & k_1 = 1 \end{array} \qquad K = \begin{bmatrix} 1 & 2 \end{bmatrix}$$



Lecture 4.2 – Pole Placement

- Pick the control gains such that the eigenvalues (poles) of the closed loop system match the desired eigenvalues

$$\chi_{A-BK}(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0$$

$$\varphi(\lambda) = \prod_{i=1}^{n}(\lambda - \lambda_i) = \lambda^n + b_{n-1}\lambda^{n-1} + \ldots + b_1\lambda + b_0$$

$$\begin{aligned} \lambda^{n-1}: &\qquad a_{n-1}(K) = b_{n-1} \\ &\vdots \\ \lambda^0: &\qquad a_0(K) = b_0 \end{aligned}$$

Questions

- Is this always possible? *No!*
- How should we pick the eigenvalues? *Mix of art and science*
- Do we have to compute large determinants? *No!*

  In MATLAB:

```
>> P=[lambda1,lambda2,lambda3,…];
>> K=place(A,B,P);
```

Example

$$\dot{x} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u$$

$$u = -Kx = \begin{bmatrix} k_1 & k_2 \end{bmatrix} x \Rightarrow A - BK = \begin{bmatrix} 2 - k_1 & -k_2 \\ 1 - k_1 & 1 - k_2 \end{bmatrix}$$

$$\chi_{A-BK}(\lambda) = \begin{vmatrix} \lambda - 2 + k_1 & k_2 \\ -1 + k_1 & \lambda - 1 + k_2 \end{vmatrix}$$

$$= \lambda^2 + \lambda(-3 + k_1 + k_2) + 2 - k_1 - k_2$$

Example

$$\chi_{A-BK}(\lambda) = \lambda^2 + \lambda(-3 + k_1 + k_2) + 2 - k_1 - k_2$$

$$\varphi(\lambda) = (\lambda + 1)^2 = \lambda^2 + 2\lambda + 1$$

$$\lambda^1: \quad -3 + k_1 + k_2 = 2 \Rightarrow \quad k_1 + k_2 = 5$$

$$\lambda^0: \quad 2 - k_1 - k_2 = 1 \Rightarrow \quad k_1 + k_2 = 1$$

???

- What's at play here is a lack of "controllability", i.e., the effect of the input is not sufficiently rich to influence the system enough
- More on controllability later…

How to Pick Eigenvalues?

- Assuming that we have sufficient control authority to do pole placement, how should the desired eigenvalues be selected?
- No clear-cut answer
- Some things to keep in mind:

(complex-conjugate pairs)

$$\mathrm{Im}(\lambda_i) \neq 0 \Rightarrow \exists \lambda_j : \mathrm{Im}(\lambda_j) = -\mathrm{Im}(\lambda_i), \;\; \mathrm{Re}(\lambda_j) = \mathrm{Re}(\lambda_i)$$

$$\mathrm{Re}(\lambda_i) < 0, \;\; \forall \lambda_i \;\; \text{(stability)}$$

$$\mathrm{Im}(\lambda_i) \neq 0 \Rightarrow \;\; \text{oscillations}$$

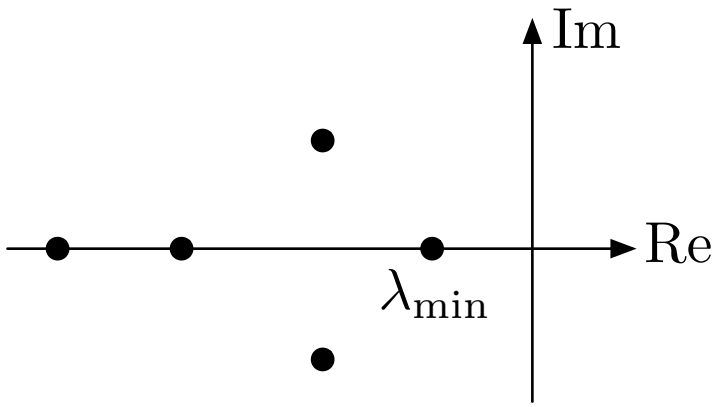
Rate of convergence

Rate of Convergence

$$\dot{x} = (A - BK)x, \quad \mathrm{Re}(\lambda) < 0, \ \forall \lambda \in \mathrm{eig}(A - BK)$$

- The "smallest" eigenvalue dominates the convergence rate

$$\lambda_{\min} = \mathrm{argmin}_{\lambda} |\mathrm{Re}(\lambda)|$$



- But, the bigger the eigenvalues the bigger the control gains/signals

In MATLAB

```
% System matrices
A=[2,0;1,-1]; B=[1;1];

% Let's pick our favorite poles
P=[-0.5+1i,-0.5-1i];

% Pole placement
K=place(A,B,P);

% Compute the solution
x=[1;1]; t=0; tf=5; dt=0.01;

while (t<tf);
    x=x+dt.*(A-B*K)*x;
    t=t+dt;
end;
```

Lecture 4.3 – Controllability

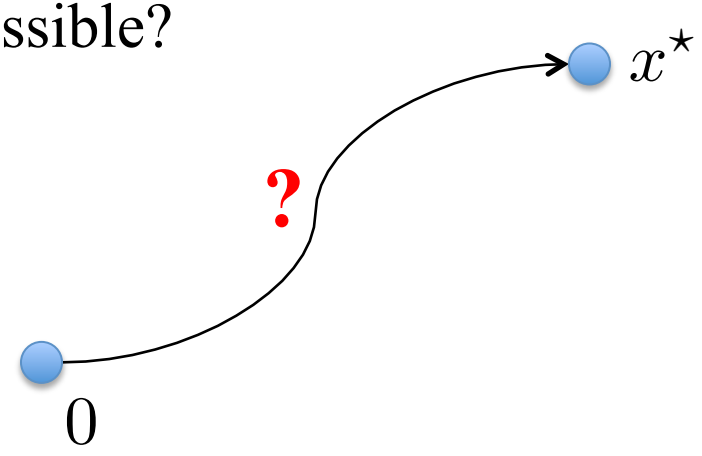
- When can we place the eigenvalues however we want using state feedback?
- When is the B matrix (the actuator configuration) rich enough so that we can make the system do whatever we want it to do?
- The answer revolves around the concept of *controllability*!

A Modest Example

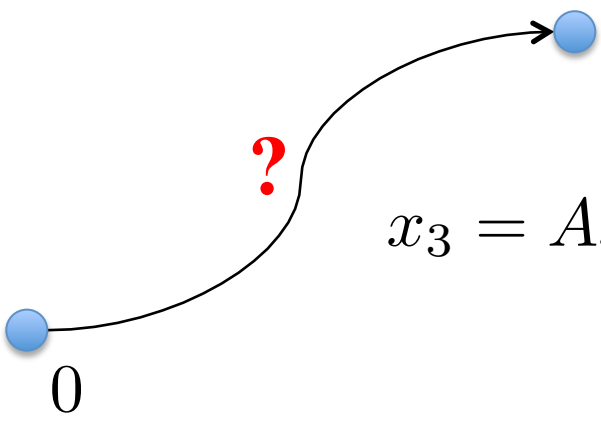
- Given a discrete-time system

$$x_{k+1} = Ax_k + Bu_k, \quad x_0 = 0$$

- We would like to drive this system in *n* steps to a particular target state $x^\star$
- Is this possible?



A Modest Example



$$x_1 = Ax_0 + Bu_0 = Bu_0$$

$$x_2 = Ax_1 + Bu_1 = ABu_0 + Bu_1$$

$$x_3 = Ax_2 + Bu_2 = A^2Bu_0 + ABu_1 + Bu_2$$

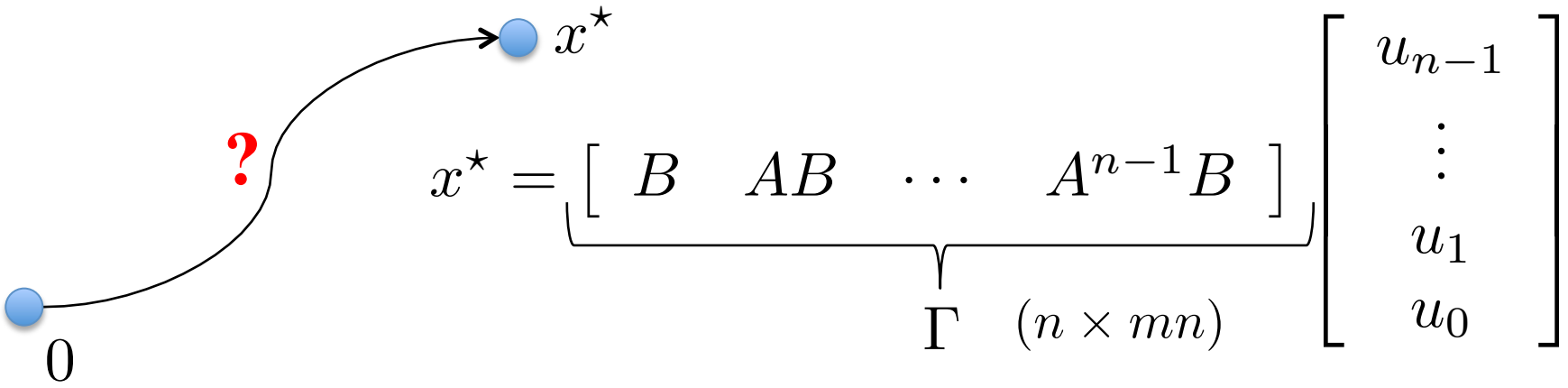
$$\vdots$$

$$x_n = A^{n-1}Bu_0 + \ldots + Bu_{n-1}$$

We want to solve

$$x^\star = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \begin{bmatrix} u_{n-1} \\ \vdots \\ u_1 \\ u_0 \end{bmatrix}$$

A Modest Example



- This is possible for any target state if and only if

$$\text{rank}(\Gamma) = n$$

- Turns out, this generalizes quite nicely…

Controllability – Theorem 1

$$\dot{x} = Ax + Bu, \quad x \in \Re^n$$

Definition: The system is *completely controllable* (CC) if it is possible to go from any initial state to any final state.

$$\Gamma = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \longleftarrow$$ controllability matrix

CC Theorem 1: The system is CC if and only if

$$\text{rank}(\Gamma) = n$$

rank(M)=# linearly independent rows or columns in M

Two Systems

$$\dot{x} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u \qquad n = 2 \qquad \Gamma = [B \; AB]$$

pole-placement not possible

not CC!

$$AB = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \qquad \text{rank}(\Gamma) = 1$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad n = 2 \qquad \Gamma = [B \; AB]$$

pole-placement possible

CC!

$$AB = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \text{rank}(\Gamma) = 2$$

Controllability – Theorem 2

$$u = -Kx, \quad \dot{x} = (A - BK)x$$

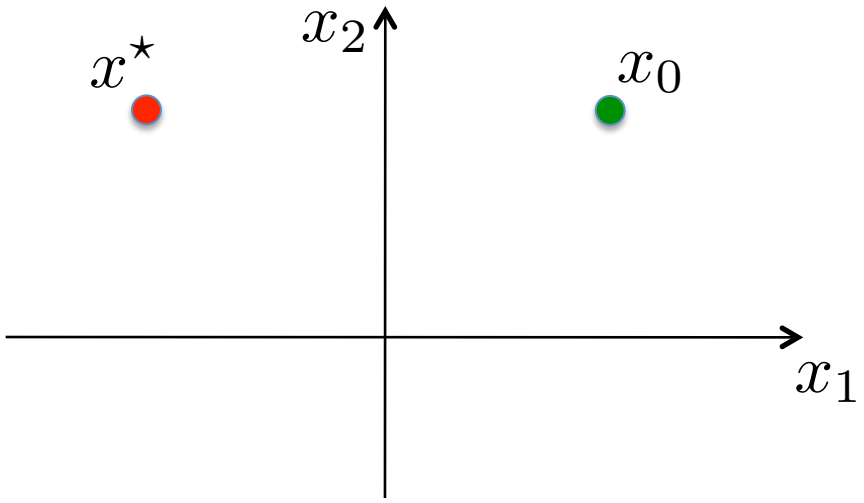
CC Theorem 2: Pole-placement to arbitrary eigenvalues is possible if and only if the system is CC!

Point-to-Point vs. Trajectories

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

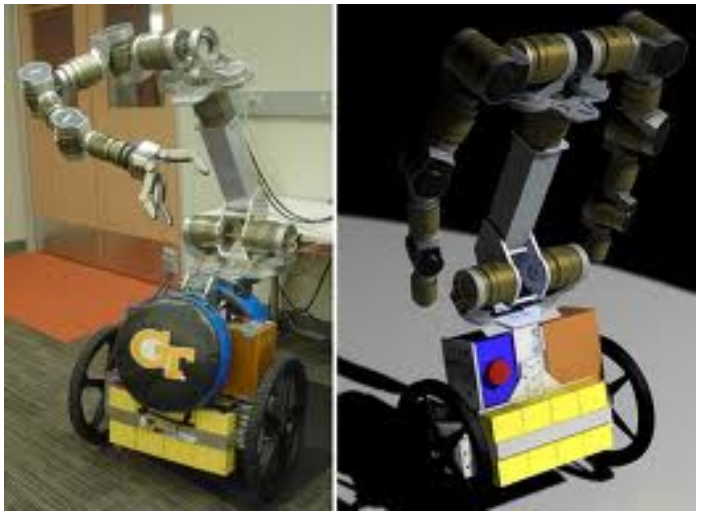
In MATLAB:

```
>> G=ctrb(A,B);
>> rank(G)
   ans: 2
```

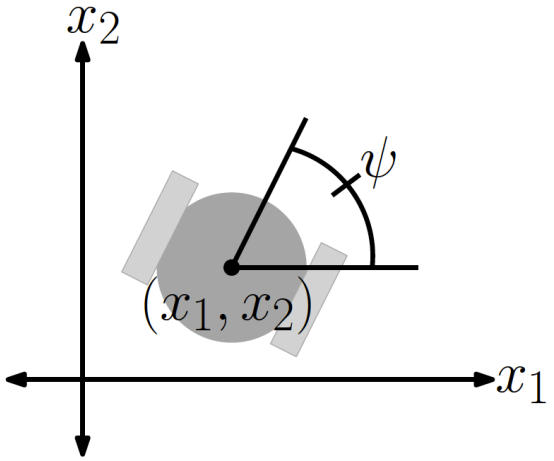


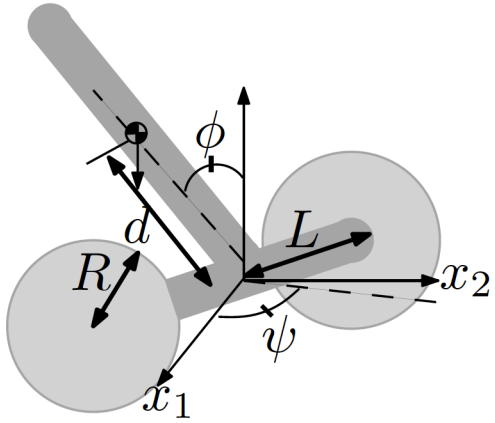
Lecture 4.4 – The Segway Robot

- Let's unleash our newfound powers on a complicated robotic system



Unicycle + Inverted Pendulum + …





The Base:

$$\dot{x}_1 = v\cos\psi$$
$$\dot{x}_2 = v\sin\psi$$
$$\dot{\psi} = \omega$$

The "Pendulum":

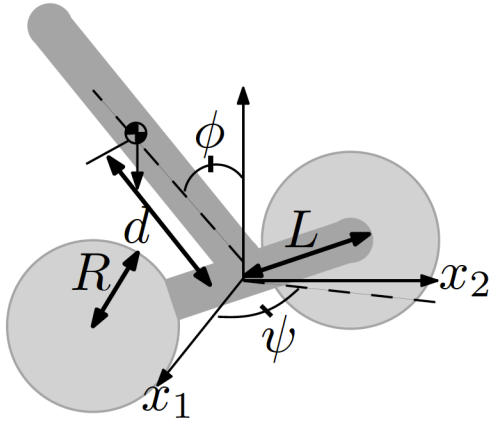
$$\phi,\ \dot{\phi}$$

Inputs: $\tau_L,\ \tau_R$

Extra states: $v,\ \omega$

Unicycle + Inverted Pendulum + …

$$x = [x_1 \quad x_2 \quad v \quad \psi \quad \dot{\psi} \quad \phi \quad \dot{\phi}]^T$$
$$u = [\tau_L \quad \tau_R]^T$$
$$\dot{x} = f(x, u)$$
$$= [v\cos\psi \quad v\sin\psi \quad \dot{v} \quad \dot{\psi} \quad \ddot{\psi} \quad \dot{\phi} \quad \ddot{\phi}]^T$$



$$3(m_w + m_b)\dot{v} - m_b d\cos\phi\ddot{\phi} + m_b d\sin\phi(\dot{\phi}^2 + \dot{\psi}^2) = -\frac{1}{R}(\tau_L + \tau_R)$$

LINEARIZE!

$$((3L^2 + \frac{1}{2R^2})m_w + m_b d^2\sin^2\phi + I_2)\ddot{\psi} + m_b d^2\sin\phi\cos\phi\dot{\psi}\dot{\phi} = \frac{L}{R}(\tau_L - \tau_R)$$

$$m_b d\cos\phi\dot{v} + (-m_b d^2 - I_3)\ddot{\phi} + m_b d^2\sin\phi\cos\phi\dot{\phi}^2 + m_b g d\sin\phi = \tau_L + \tau_R$$

Linearization

- Linearize this mess around (x,u)=(0,0), and plug in the physical parameters for the segway robot:

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

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.16 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 72.49 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1.67 & -1.67 \\ 0 & 0 \\ 0.029 & -0.029 \\ 0 & 0 \\ -24.15 & -24.1514 \end{bmatrix}$$

Controllability?

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

ans =

     6
```

Not CC!

- We already know that the unicycle dynamics gets messed up when linearized.
- And if we can control v and ω, that should be "enough", so let's simply remove the unicycle part

A Smaller System

$$x = [v \quad \omega \quad \phi \quad \dot{\phi}]^T, \quad u = [\tau_L \quad \tau_R]^T$$

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

$$A = \begin{bmatrix} 0 & 0 & 2.16 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 72.49 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -1.67 & -1.67 \\ 0.029 & -0.029 \\ 0 & 0 \\ -24.15 & -24.15 \end{bmatrix}$$

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

ans =

     4
```

CC!

One Last Twist

- We do not want to stabilize to (v,ω)=(0,0)
- Let

$$\tilde{x} = x - \begin{bmatrix} v_d & \omega_d & 0 & 0 \end{bmatrix}^T = x - \delta$$

- The dynamics become

$$\dot{\tilde{x}} = \dot{x} - \dot{\delta} = \dot{x} = Ax + Bu = A(x - \delta) + Bu + A\delta$$

- Luckily,

$$A\delta = 0 \;\Rightarrow\; \dot{\tilde{x}} = A\tilde{x} + Bu$$

Pole-Placement

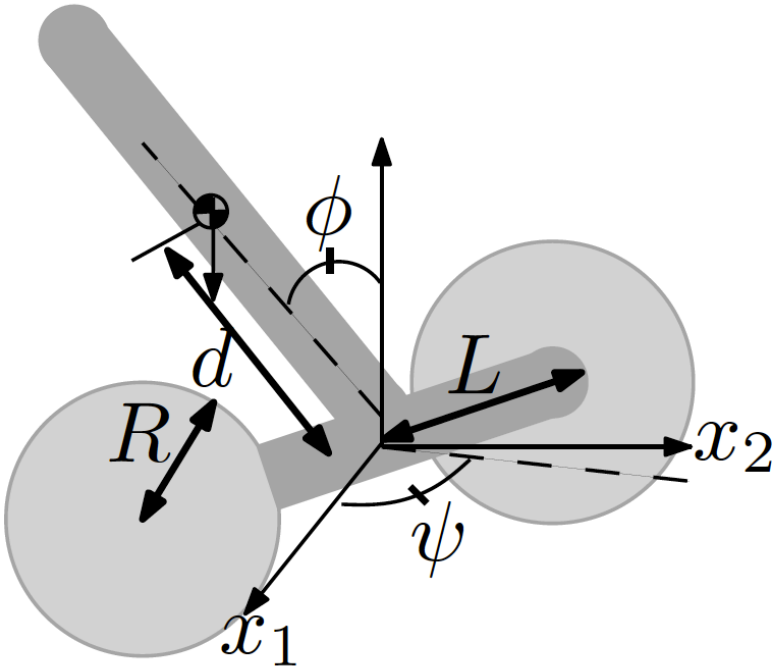
- We have a CC system that we wish to stabilize
- After some tweaking, the following eigenvalues seem to behave well:

$$\lambda_1 = -19,\ \lambda_2 = -7.5,\ \lambda_3 = -5,\ \lambda_4 = -0.3$$

$$u = -K\tilde{x} \ \Rightarrow\ \dot{\tilde{x}} = (A - BK)\tilde{x}$$

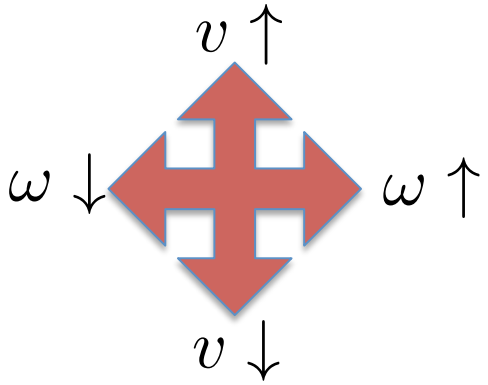
```
>> P=[-19,-7.5,-5,-0.3];

>> K=place(A,B,P);
```

Now, Let's Do It!



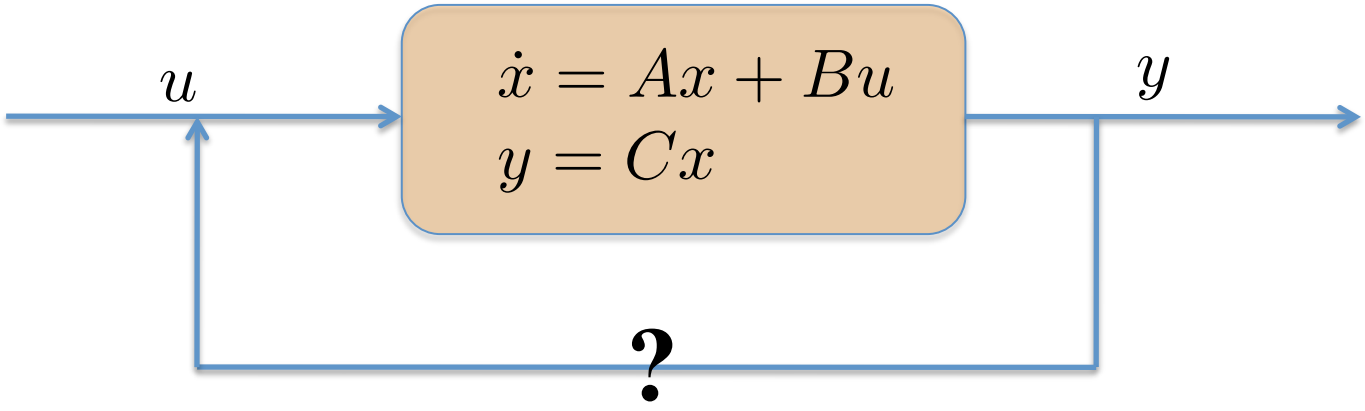
$$\kappa = \frac{\omega}{v}$$

curvature rather than pose



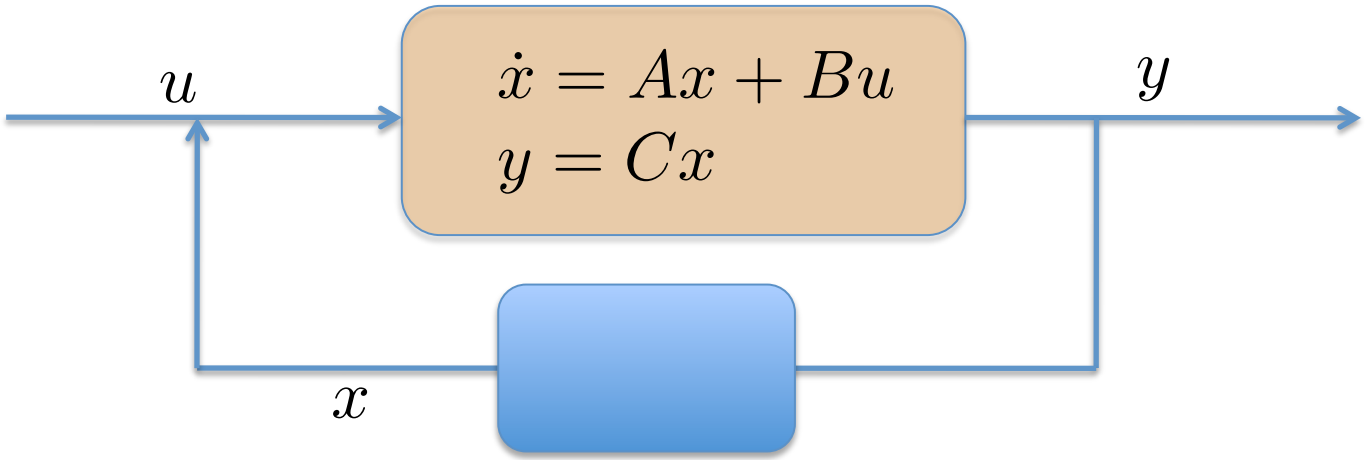
Lecture 4.5 – Observers

- We now know how to design rather effective controllers using state feedback.
- *But what about y???*



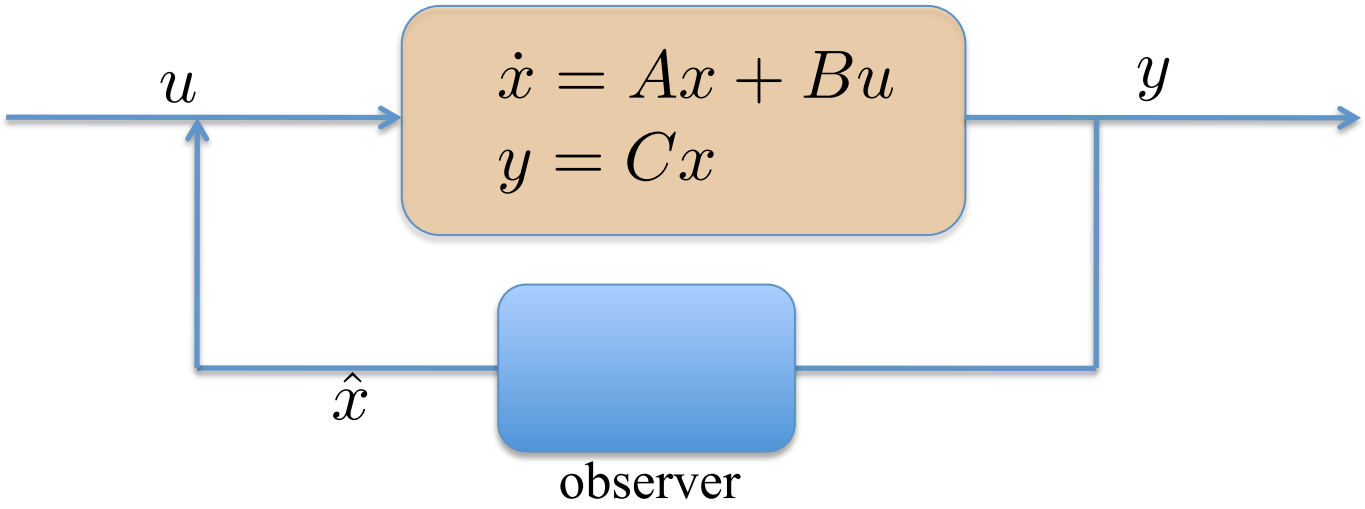
Lecture 4.5 – Observers

- We now know how to design rather effective controllers using state feedback.
- *But what about y???*



Lecture 4.5 – Observers

- We now know how to design rather effective controllers using state feedback.
- *But what about y???*



The Predictor-Corrector

$$\dot{x} = Ax$$
$$y = Cx$$

First idea: Make a copy of the system

$$\dot{\hat{x}} = A\hat{x}$$ **predictor**

Second idea: Add a notion of how wrong your estimate is to the model

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

corrector

"Luenberger" observer

Picking the Observer Gain

$\dot{x} = Ax + Bu$

$u = Kx$

- What we want to stabilize (drive to zero) is the *estimation error*, i.e., the difference between the actual state and the estimated state $e = x - \hat{x}$

$$\dot{e} = \dot{x} - \dot{\hat{x}} = Ax - A\hat{x} - L(\overbrace{y}^{Cx} - C\hat{x})$$

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

- Just pick L such that the eigenvalues to A-LC have negative real part!!

$\dot{e} = (A - LC)e$ same as $\dot{x} = (A - BK)x$

Pole-Placement, Again

$$\dot{e} = (A - LC)e$$

- Want

$$\mathrm{Re}(\mathrm{eig}(A - LC)) < 0$$

- We already know how to do this - Pole-placement!

Example

$$\dot{x} = \begin{bmatrix} -1 & 2 \\ 0 & -2 \end{bmatrix} x, \quad y = \begin{bmatrix} 1 & -1 \end{bmatrix} x$$

$$\dot{e} = (A - LC)e = \left(\begin{bmatrix} -1 & 2 \\ 0 & -2 \end{bmatrix} - \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} \right) e$$

$$= \begin{bmatrix} -1 - L_1 & 2 + L_1 \\ -L_2 & -2 + L_2 \end{bmatrix} e$$

Example

$$\dot{e} = \begin{bmatrix} -1 - L_1 & 2 + L_1 \\ -L_2 & -2 + L_2 \end{bmatrix} e$$

$$\chi_{A-LC}(\lambda) = \begin{vmatrix} \lambda + 1 + L_1 & -2 - L_1 \\ L_2 & \lambda + 2 - L_2 \end{vmatrix}$$

$$= \lambda^2 + \lambda(3 + L_1 - L_2) + 2 + 2L_1 + L_2$$

$$\varphi(\lambda) = (\lambda + 1)^2 = \lambda^2 + 2\lambda + 1$$

$$\begin{cases} L_1 = -2/3 \\ L_2 = 1/3 \end{cases}$$

$3 + L_1 - L_2 = 2$

$2 + 2L_1 + L_2 = 1$

$5 + 3L_1 = 3$

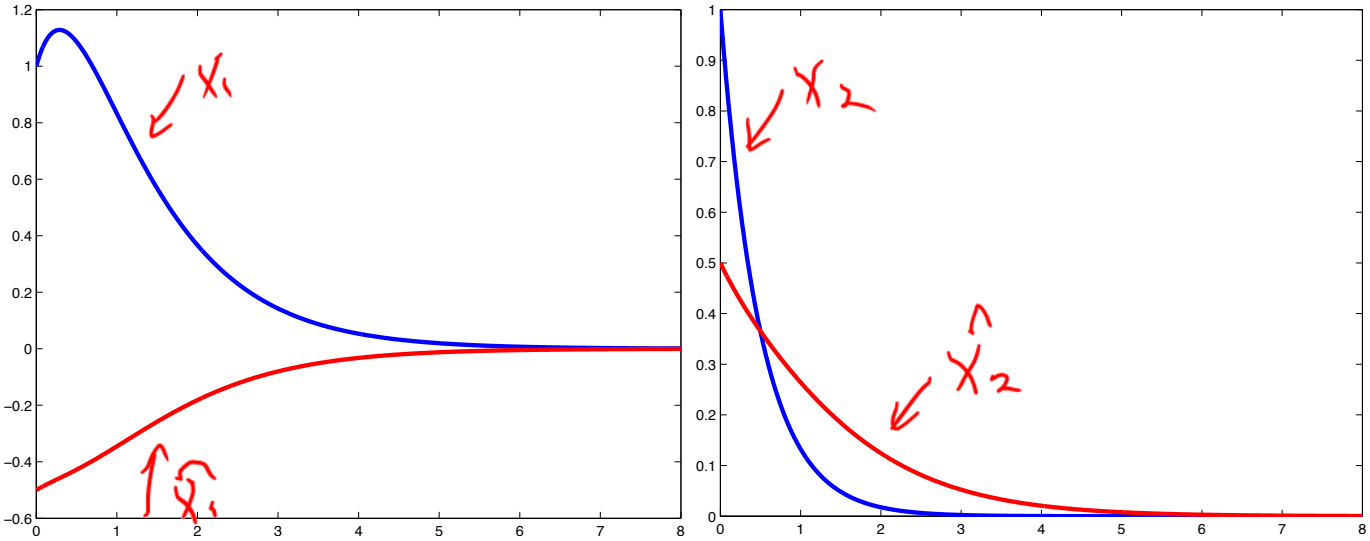
$L_1 = -\frac{2}{3}$

$L_2 = \frac{1}{3}$

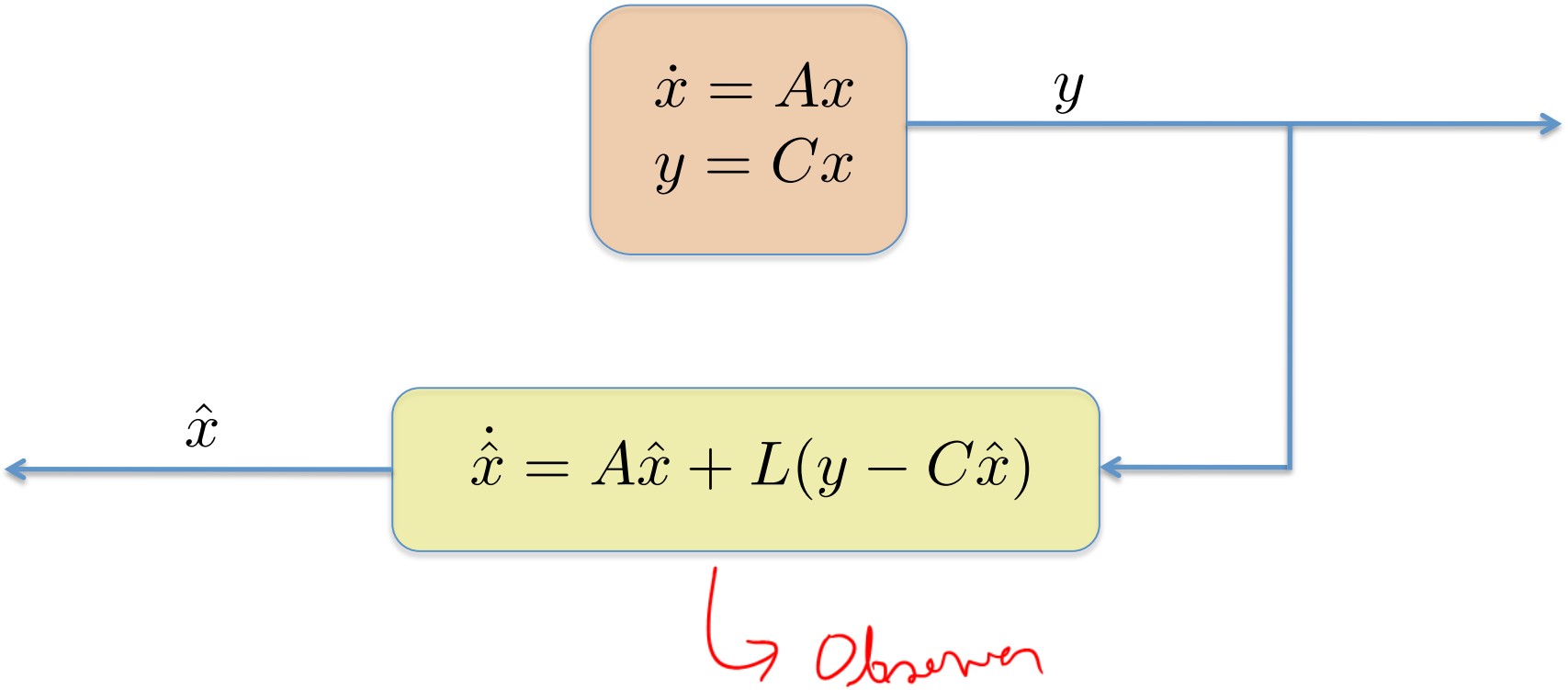
Example

$$\dot{\hat{x}} = A\hat{x} + \frac{1}{3}\begin{bmatrix} -2 \\ 1 \end{bmatrix}(y - C\hat{x})$$

Observer dynamics



The Block-Diagram



Does This Always Work?

- **No!**

- Next Lecture: We need to redo what we did for control design (***controllability***) for observer design (***observability***)

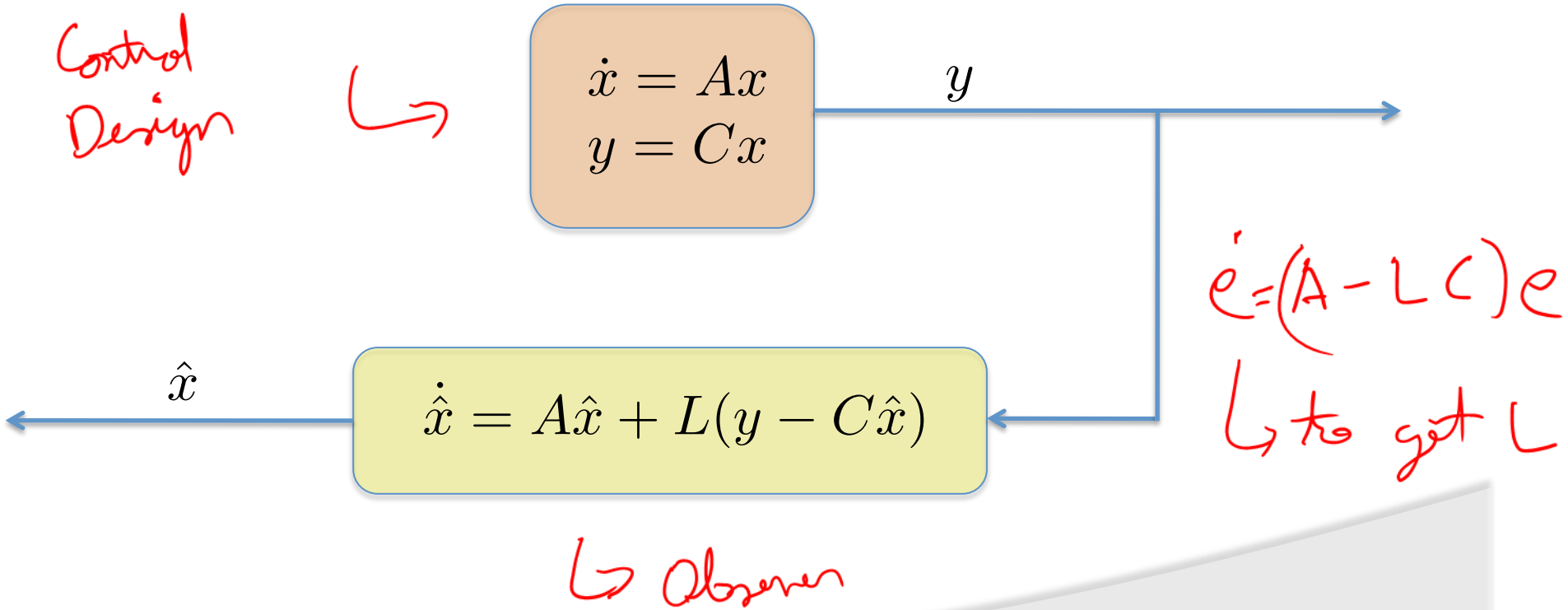
A = physics

B = actuation

y = sensor

Lecture 4.6 – Observability

- Need to redo what we did for control design to understand when we can recover the state from the output



A Modest Example (Again)

- Given a discrete time system without inputs

$$x_{k+1} = Ax_k$$
$$y_k = Cx_k$$

- Can we recover the initial condition by collecting *n* output values?

$$y_0 = Cx_0$$
$$y_1 = Cx_1 = CAx_0$$
$$\vdots$$
$$y_{n-1} = CA^{n-1}x_0$$

The Observability Matrix

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{\Omega} x_0$$

Controllability Matrix

$\text{rank}(\Omega) = n$

Ω → Similar to Γ

- The initial condition can be recovered from the outputs when the so-called observability matrix has full rank.

Observability – Theorem 1

$$\dot{x} = Ax, \quad x \in \Re^n$$
$$y = Cx$$

Definition: The system is *completely observable* (CO) if it is possible to recover the initial state from the output.

$$\Omega = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

← observability matrix

Same as CC

CO Theorem 1: The system is CO if and only if

$$\text{rank}(\Omega) = n$$

rank(M)=# linearly independent rows or columns in M

Observability – Theorem 2

$$\dot{\hat{x}} = A\hat{x} + L(y - C\hat{x}), \quad \dot{e} = (A - LC)e$$

CO Theorem 2: Pole-placement to arbitrary eigenvalues is possible if and only if the system is CO!

Putting It All Together

- We are very close to being able to design the controllers as if we had x while all we really have is y (and hence an estimate of x).
- Next lecture – Putting it all together!

Lecture 4.7 – The Separation Principle

- We have lots of really good building blocks:
  - Controllability ← $\dot{x} = (A - BK)x$
  - Observability ← $\dot{e} = (A - LC)e$
  - State feedback
  - Observers
  - Pole-placement

- Now, how do we put everything together?
- Answer: The wonderful ***Separation Principle***

A Game Plan

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

Assume CC and CO

Step 1: Design the state feedback controller as if we had x (which we don't)

closed-loop dynamics

$$u = -Kx \;\Rightarrow\; \dot{x} = (A - BK)x \longleftarrow \text{what we design for}$$

$$u = -K\hat{x} \longleftarrow \text{what we actually have}$$

Step 2: Estimate x using an observer (that now also contains u)

predictor corrector

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

$$\Rightarrow \; \dot{e} = (A - LC)e$$

$e = x - \hat{x}$

Does This Work?

- Want both x and e to be stabilized
- Analyze their joint dynamics:

$$\dot{x} = Ax - BK\hat{x} = Ax - BK(x - e) = (A - BK)x + BKe$$

$$e = x - \hat{x}$$

$$\dot{e} = (A - LC)e$$

Eigenvalues < 0

- Or, when viewed together

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}$$

- The strategy works if and only if this is an asymptotically stable system!

The Separation Principle

Eig are all dependant to diagonal values

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix}}_{M} \begin{bmatrix} x \\ e \end{bmatrix}$$

- This is an (upper) triangular block-matrix. Its eigenvalues are given by the eigenvalues of the diagonal blocks!

$$\chi_M(\lambda) = \chi_{A-BK}(\lambda)\chi_{A-LC}(\lambda)$$

- If we haven't messed up in the design, we have

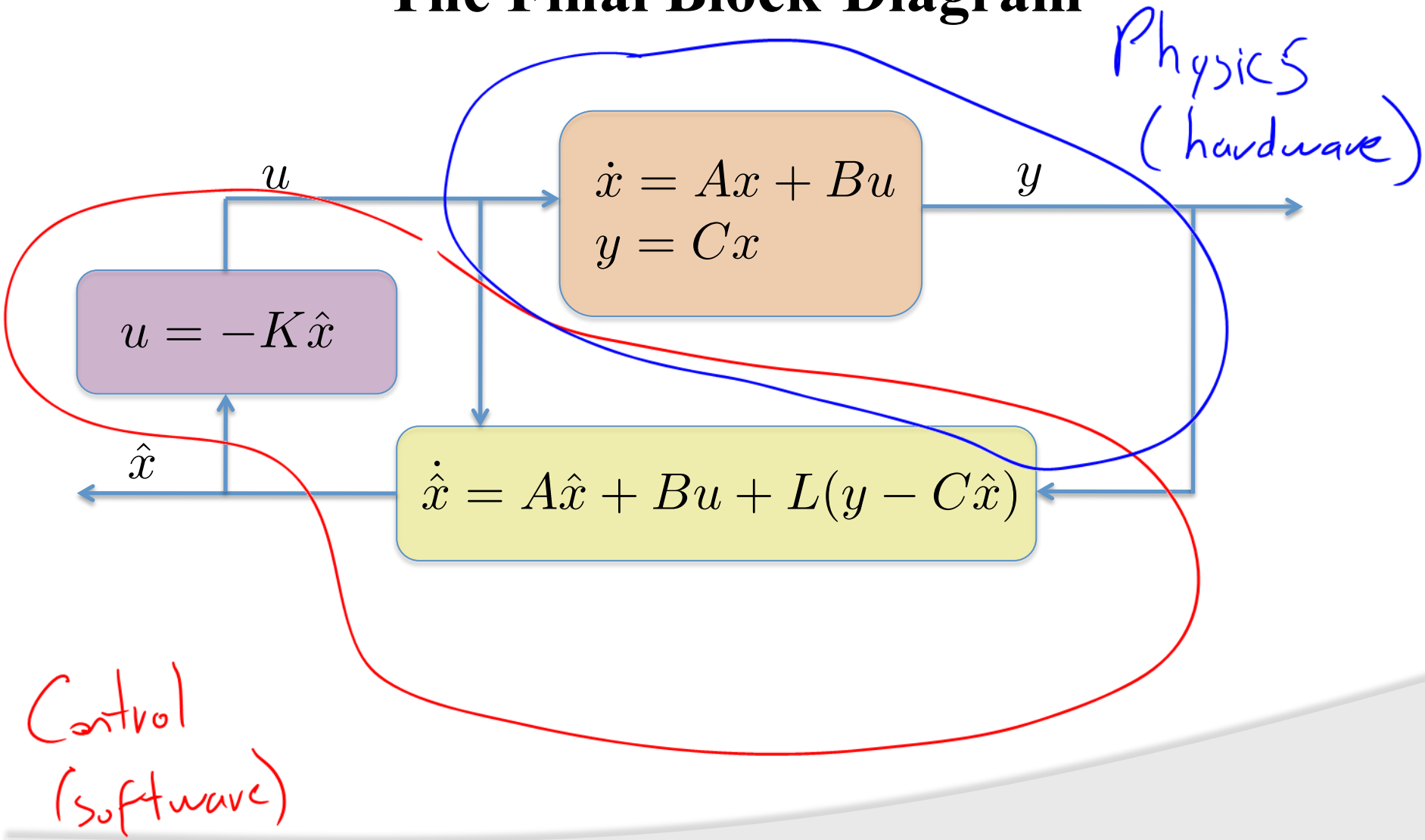
$$\text{Re}(\text{eig}(A - BK)) < 0, \quad \text{Re}(\text{eig}(A - LC)) < 0$$

- Everything works!!

The Separation Principle

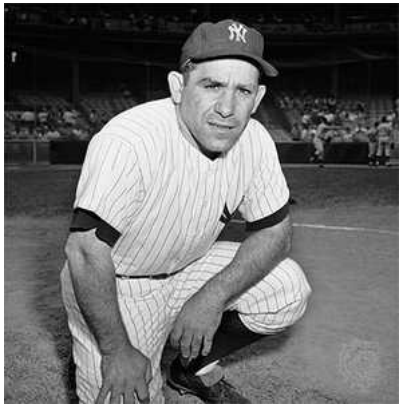
- This remarkable fact is known as the ***Separation Principle***!
- Means that we can design controllers as if we have x
- Can design observers independent of the control actions
- Control and observer designs can be separated!

The Final Block-Diagram



Lecture 4.8 – Practical Considerations

- The separation principle means that we can decouple control and observer design (in theory)



"*In theory, theory and practice are the same. In practice they are not.*" -Yogi Berra

Eigenvalue Selection

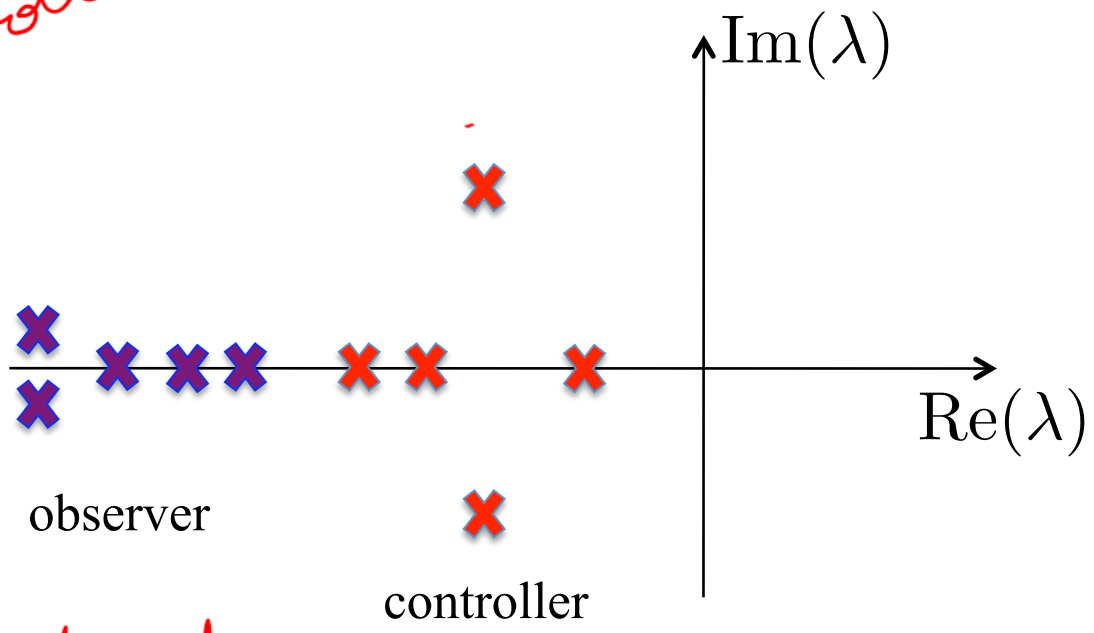
- Typically, the observer should be faster than the controller, since the controller won't do anything useful until the state estimate is close.
- Note: Large observer eigenvalues give large observer gains:
  - Not a problem – The observer is a software construct!

Observer needs to converge in order for the controller to be useful

Use Large Eingenvalues

Eigenvalue Selection

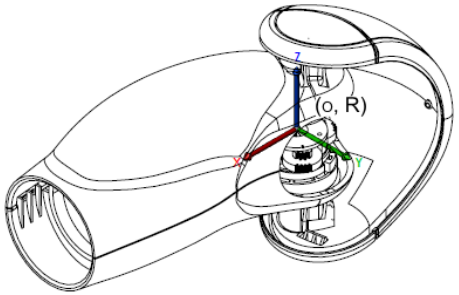
Observer eigenvalue should be larger than the controller.



Smallest Eigenvalue for observer > than largest controller eigenvalue

Humanoid Robot

Aldebaran Nao



torque constant

$$\ddot{\theta} = \frac{1}{J}(Ki - b\dot{\theta})$$

moment of inertia, current, friction coefficient

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -b/J \end{bmatrix} x + \begin{bmatrix} 0 \\ K/J \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

$u = i$

$x_1 = \theta$

$x_2 = \dot{\theta}$

$\dot{x}_1 = \dot{\theta} = x_2$

$\dot{x}_2 = \ddot{\theta} = \frac{1}{J}(Ki - b\dot{\theta})$

$y = \theta$

CC & CO!!

↳ $\text{rank}(\Gamma) = n$

$\text{rank}(\mathcal{L}) = n$

Reference Tracking

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

$$e = \begin{bmatrix} \theta - \theta_d \\ \dot{\theta} \end{bmatrix}$$

desired angle

$e = 0 \rightarrow \theta = \theta_d$, $\dot{\theta} = 0$

↳ what we want

$$\dot{e} = Ax + Bu = Ae + A\begin{bmatrix} \theta_d \\ 0 \end{bmatrix} + Bu = Ae + Bu$$

$$y = Cx = Ce + C\begin{bmatrix} \theta_d \\ 0 \end{bmatrix} = Ce + \theta_d$$

$\dot{e} = \begin{bmatrix} \dot{\theta} - \dot{\theta}_d \\ \ddot{\theta} \end{bmatrix} \rightarrow 0$ (doesn't change)

$\dot{e} = \dot{x}$

$$u = -K\hat{e}$$
$$\dot{\hat{e}} = A\hat{e} + Bu + L(y - C\hat{e} - \theta_d)$$

$\hat{e}$ not e

output - what the output would have been

Beyond Pole Placement

- The design structure is quite general (does not need to be based on pole placement)

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

To design without picking Eigenvalues

- LQ Optimal Control: $\min \int_0^\infty (x^T Q x + u^T R u) dt$

$Q \geq 0$

$R > 0$

$$K = R^{-1} B^T P$$

$$0 = A^T P + PA + Q - PBR^{-1}B^T P$$

Beyond Pole Placement

- The design structure is quite general (does not need to be based on pole placement)

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

- The Kalman Filter:

$$\dot{x} = Ax + Bu + D_I v_I$$
$$y = Cx + D_O v_O$$

$$\min \lim_{t\to\infty} \|E(x(t) - \hat{x}(t))\|^2$$

$$L = PC^T(D_O D_O^T)^{-1}$$

$$0 = A^T P + PA + D_I D_I^T - PC^T(D_O D_O^T)^{-1} CP$$

Next Module

- But, the real world (especially for robots) is more complex than a simple LTI system
- **HYBRID SYSTEMS!**