Georgia Institute of Technology

Dr. Magnus Egerstedt
Professor
School of Electrical and
Computer Engineering

Control of Mobile Robots

Module 3
Linear Systems

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

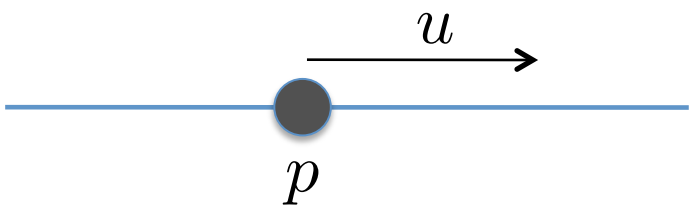
Lecture 3.1 – A Simple Robot

- So far we have seen
  - Controls Basics
  - Mobile Robots
- Need to be a bit more systematic in our discussion
- Rich class of models:
  - General enough
  - Simple enough
  - Expressive enough
  - Relevant enough

LINEAR SYSTEMS!!

"Controlling" a Point Mass

- Given a point mass on a line whose acceleration is directly controlled:



$$\ddot{p} = u$$

- Want to write this on a compact/general form

$$\begin{aligned} x_1 &= p \\ x_2 &= \dot{p} \end{aligned} \quad \Rightarrow \quad \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned}$$

On State Space Form

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = u$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Store these variables in a single, 2-dimensional "state" variable

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ u \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = p = x_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

On State Space Form

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ u \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = p = x_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

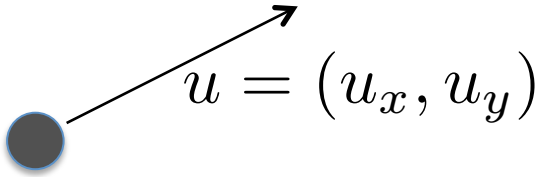
- Or, even more generally

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

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

A 2D Point-Mass

$$\ddot{p}_x = u_x$$
$$\ddot{p}_y = u_y$$



$$u = (u_x, u_y)$$

$$p = (p_x, p_y)$$

$$x_1 = p_x$$
$$x_2 = \dot{p}_x$$
$$x_3 = p_y$$
$$x_4 = \dot{p}_y$$
$$u_1 = u_x$$
$$u_2 = u_y$$
$$y_1 = p_x$$
$$y_2 = p_y$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

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

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

LTI Systems

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

- This is a so-called ***LTI*** (Linear Time-Invariant) System on State-Space form!

$$\begin{cases} x \in \Re^n \\ u \in \Re^m \\ y \in \Re^p \end{cases} \Rightarrow \begin{array}{l} A: \ n \times n \\ B: \ n \times m \\ C: \ p \times n \end{array}$$

$$\underset{n \times 1}{\dot{x}} = \underset{(n \times n)(n \times 1)}{\underset{n \times 1}{A \ x}} + \underset{(n \times m)(m \times 1)}{\underset{n \times 1}{B \ u}}$$

$$\underset{p \times 1}{y} = \underset{(p \times n)(n \times 1)}{\underset{p \times 1}{C \ x}}$$

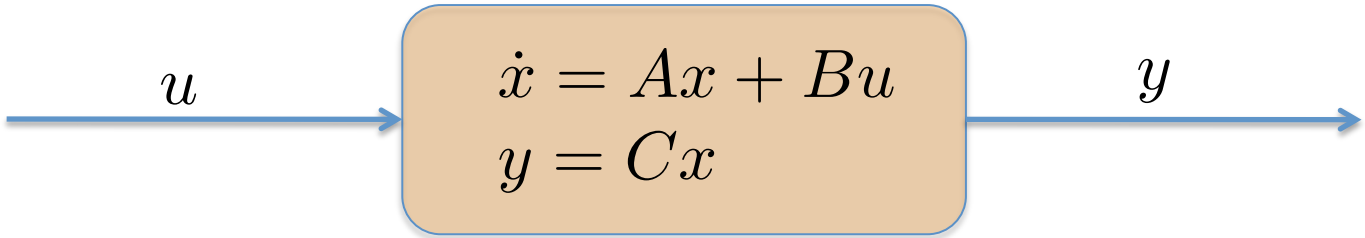
Lecture 3.2 – State-Space Models

- The general LTI model is

x – state

y – output

u – input



- Main Question: How should the input be selected?
- But First: How can such systems be understood? And where do they come from?

Example 1: The Car Model

- Recall our old friend, the car model

$$\dot{v} = \frac{c}{m}u - \gamma v$$

- If we care about/can measure the velocity:

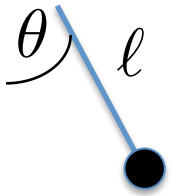
$$\dot{x} = Ax + Bu$$
$$y = Cx,$$
$$A = -\gamma,\ B = \frac{c}{m},\ C = 1$$

- If we care about/can measure the position we have the same general equation with different matrices:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\gamma \end{bmatrix},\ B = \begin{bmatrix} 0 \\ c/m \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Example 2: Pendulum

- Newton's 2nd tells us that



Not linear!

$$\ddot{\theta} = -\frac{g}{\ell}\sin(\theta) + cu$$

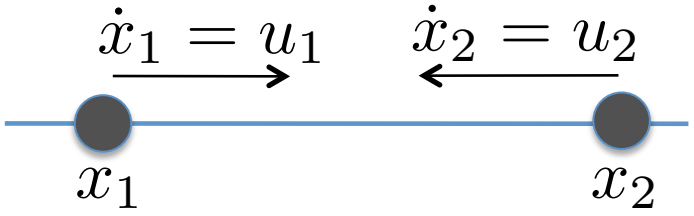
$$\theta \ll 1 \;\Rightarrow\; \sin\theta \approx \theta$$

- For small angles we get $\dot{x} = Ax + Bu,\ y = Cx$

$$A = \begin{bmatrix} 0 & 1 \\ -g/\ell & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ c \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Example 3: Two Simple "Robots"

- Consider two "robots" on a line



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

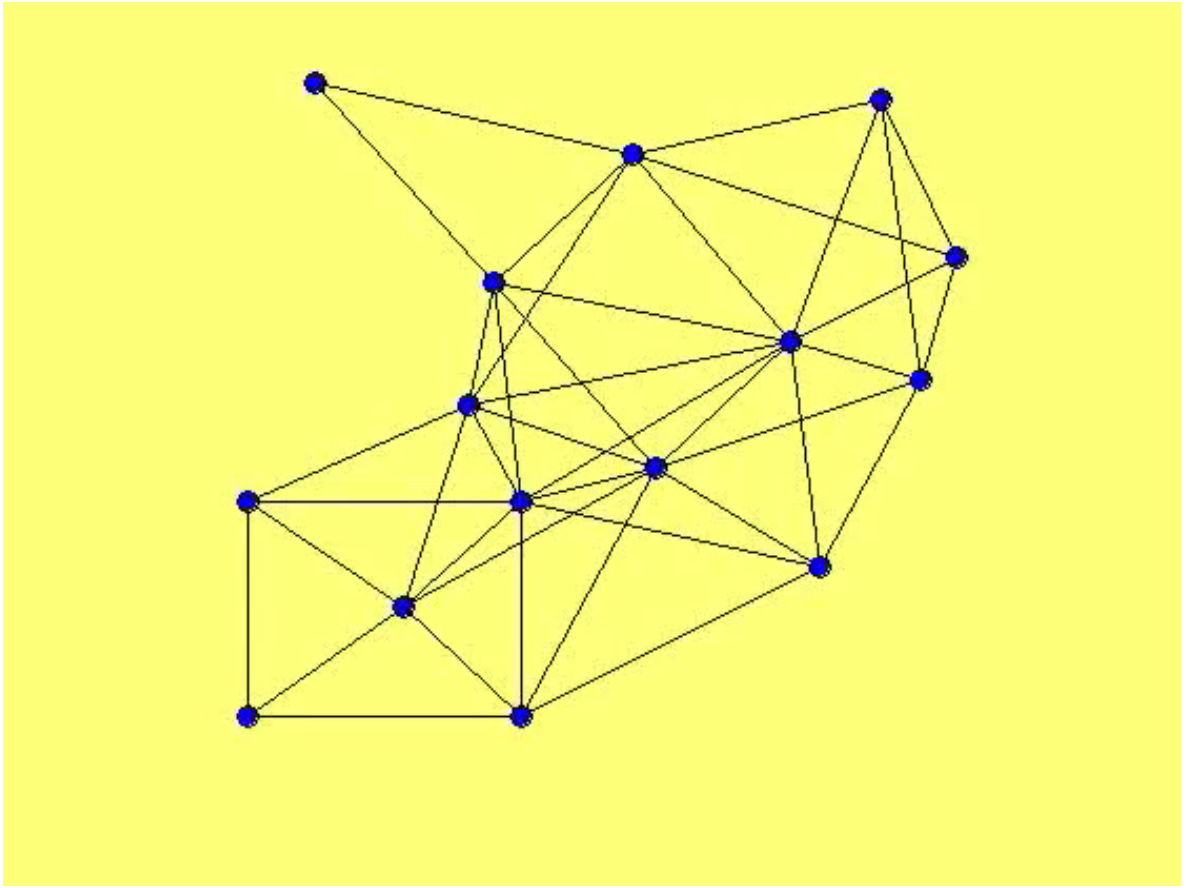
$$(A = 0)$$

- The *Rendezvous Problem*: Have them meet at the same location
- Idea: Have them aim towards each other:

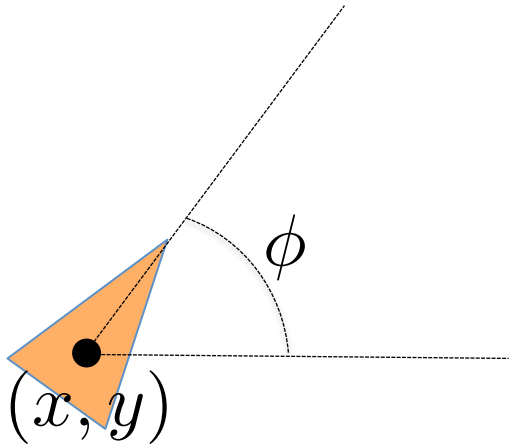
$$\begin{cases} u_1 = x_2 - x_1 \\ u_2 = x_1 - x_2 \end{cases} \qquad \dot{x} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} x$$

A closed-loop feedback law!

Rendezvous



Example 4: Unicycle Robot



$$\begin{aligned} \dot{x} &= v\cos\phi \quad \approx 1 \\ \dot{y} &= v\sin\phi \\ \dot{\phi} &= \omega \quad \approx \phi \end{aligned}$$

Not linear!

$$\begin{aligned} \dot{x} &= v \\ \dot{y} &= v\phi \\ \dot{\phi} &= \omega \end{aligned}$$

Still not linear!

- We need to be more systematic/ clever when it comes to generating LTI models from nonlinear systems!

Lecture 3.3 – Linearizations

"Classifying systems as linear and non-linear is like classifying objects in the Universe as bananas and non-bananas."

-unknown

Linearizations

- Given a non-linear model

$$\dot{x} = f(x, u), \quad y = h(x)$$

- We want to find a "local", linear model around an operating point

$$(x_o, u_o) \quad \rightarrow \quad (x = x_o + \delta x, \quad u = u_o + \delta u)$$

- The new equations of motion become

$$\dot{\delta x} = \dot{x} - \dot{x}_o = \dot{x} = f(x_o + \delta x, u_o + \delta u)$$

Linearizations

$$\dot{\delta x} = f(x_o + \delta x, u_o + \delta u)$$

Taylor expansion

$$= f(x_o, u_o) + \underbrace{\frac{\partial f}{\partial x}(x_o, u_o)}_{A}\delta x + \underbrace{\frac{\partial f}{\partial u}(x_o, u_o)}_{B}\delta u + \text{H.O.T}$$

$$y = h(x_o + \delta x) = h(x_o) + \underbrace{\frac{\partial h}{\partial x}(x_o)}_{C}\delta x + \text{H.O.T}$$

Assumptions:

$$f(x_o, u_o) = 0$$

$$h(x_o) = 0$$

Linearizations

$$\begin{cases} \dot{x} = f(x,u) \\ y = h(x) \\ \\ f(x_o, u_o) = 0 \\ h(x_o) = 0 \end{cases}$$

$$x = x_o + \delta x$$
$$u = u_o + \delta u$$

$$\begin{cases} \dot{\delta x} = A\delta x + B\delta u \\ y = C\delta x \\ \\ A = \frac{\partial f}{\partial x}(x_o, u_o) \\ \\ B = \frac{\partial f}{\partial u}(x_o, u_o) \\ \\ C = \frac{\partial h}{\partial x}(x_o) \end{cases}$$

Computing the Jacobians

$$x \in \Re^n,\ u \in \Re^m,\ y \in \Re^p,\ f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix},\ h = \begin{bmatrix} h_1 \\ \vdots \\ h_m \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$n \times n$$

Computing the Jacobians

$$x \in \Re^n,\ u \in \Re^m,\ y \in \Re^p,\ f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix},\ h = \begin{bmatrix} h_1 \\ \vdots \\ h_m \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \frac{\partial f_2}{\partial u_1} & \cdots & \frac{\partial f_2}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}$$

$$n \times m$$

Computing the Jacobians

$$x \in \Re^n,\ u \in \Re^m,\ y \in \Re^p,\ f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix},\ h = \begin{bmatrix} h_1 \\ \vdots \\ h_m \end{bmatrix}$$

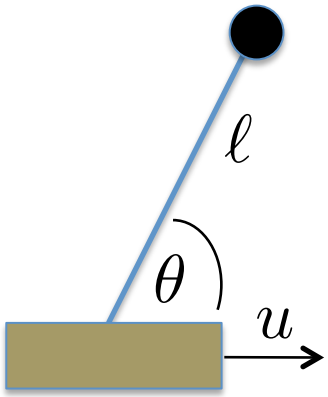
$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \frac{\partial f_2}{\partial u_1} & \cdots & \frac{\partial f_2}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}$$

$$\frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_p}{\partial x_1} & \cdots & \frac{\partial h_p}{\partial x_n} \end{bmatrix}$$

$$p \times n$$

Example: Inverted Pendulum



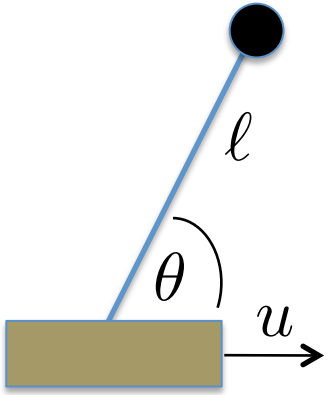
$$\ddot{\theta} = \frac{g}{\ell}\sin\theta + u\cos\theta$$

$$x_1 = \theta \qquad f(x,u) = \begin{bmatrix} x_2 \\ g/\ell\sin(x_1) + u\cos(x_1) \end{bmatrix}$$
$$x_2 = \dot{\theta} \;,$$
$$y = x_1 \qquad h(x) = x_1$$

$$(x_o, u_o) = (0,0)$$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}_{(0,0)} = \begin{bmatrix} 0 & 1 \\ g/\ell\cos(x_1) & 0 \end{bmatrix}_{(0,0)} = \begin{bmatrix} 0 & 1 \\ g/\ell & 0 \end{bmatrix}$$

Example: Inverted Pendulum



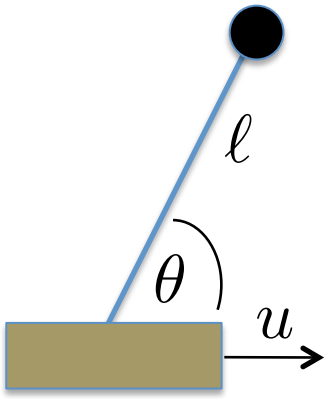
$$\ddot{\theta} = \frac{g}{\ell}\sin\theta + u\cos\theta$$

$$x_1 = \theta \qquad f(x,u) = \begin{bmatrix} x_2 \\ g/\ell\sin(x_1) + u\cos(x_1) \end{bmatrix}$$
$$x_2 = \dot{\theta} \;, \qquad$$
$$y = x_1 \qquad h(x) = x_1$$

$$(x_o, u_o) = (0,0)$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix}_{(0,0)} = \begin{bmatrix} 0 \\ \cos(x_1) \end{bmatrix}_{(0,0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Example: Inverted Pendulum



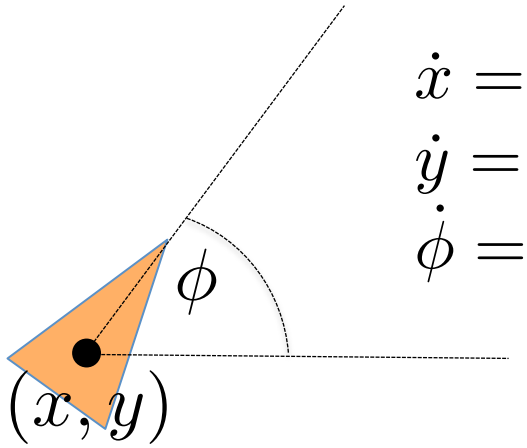
$$\ddot{\theta} = \frac{g}{\ell}\sin\theta + u\cos\theta$$

$$x_1 = \theta \qquad f(x,u) = \begin{bmatrix} x_2 \\ g/\ell\sin(x_1) + u\cos(x_1) \end{bmatrix}$$
$$x_2 = \dot{\theta} \ , \qquad$$
$$y = x_1 \qquad h(x) = x_1$$

$$(x_o, u_o) = (0,0)$$

$$C = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix}_{(0,0)} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Example: Unicycle



$$\dot{x} = v\cos\phi$$
$$\dot{y} = v\sin\phi$$
$$\dot{\phi} = \omega$$

$$\begin{cases} x_1 = x,\ x_2 = y,\ x_3 = \phi \\ y_1 = x_1,\ y_2 = x_2,\ y_3 = x_3 \\ u_1 = v,\ u_2 = \omega \\ (x_o, u_o) = (0, 0) \end{cases}$$

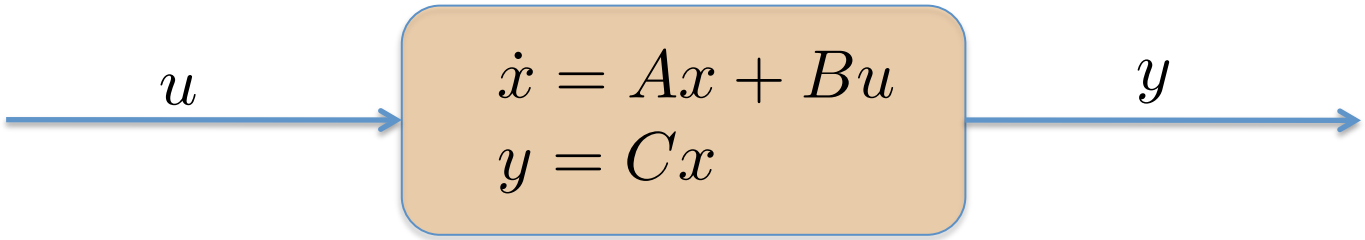
$$A = 0,\ B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\dot{x}_2 = 0 \ \text{???}$$

Punchlines

- Sometimes the linearizations give reasonable models and sometimes they do not…
- Despite the fact that they are only local approximations, they are remarkably useful (when they work…)

Lecture 3.4 – LTI Systems



- Let's figure out how such systems behave.
- Start by ignoring the input term:

$$\dot{x} = Ax$$
$$x(t_0) = x_0$$

- What is the solution to this system?

Solving the ODE

- If everything is scalar:

$$\dot{x} = ax, \quad x(t_0) = x_0 \quad \Rightarrow \quad x(t) = e^{a(t-t_0)}x_0$$

- How do we know?

$$x(t_0) = e^{a(t_0-t_0)}x_0 = e^0 x_0 = x_0$$ ✓ initial conditions

$$\frac{d}{dt}x(t) = ae^{a(t-t_0)}x_0 = ax$$ ✓ dynamics

- For higher-order systems, we just get a matrix version of this

$$\dot{x} = Ax, \; x(t_0) = x_0 \; \Rightarrow \; x(t) = e^{A(t-t_0)}x_0$$

matrix exponential

Matrix Exponentials

- The definition is just like for scalar exponentials

$$e^{at} = \sum_{k=0}^{\infty} \frac{a^k t^k}{k!}, \qquad e^{At} = \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

- Derivative:

$$\frac{d}{dt} \sum_{k=0}^{\infty} \frac{A^k t^k}{k!} = 0 + \sum_{k=1}^{\infty} \frac{k A^k t^{k-1}}{k!} = A \sum_{k=1}^{\infty} \frac{A^{k-1} t^{k-1}}{(k-1)!} = A \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

$$\frac{d}{dt} e^{At} = A e^{At}$$

Solving the Controlled Equation

- The matrix exponential plays such an important role that it has its own name: ***The State Transition Matrix***

$$e^{A(t-t_0)} = \Phi(t, t_0)$$

$$\dot{x} = Ax \;\Rightarrow\; x(t) = \Phi(t,\tau)x(\tau) \quad \begin{cases} \frac{d}{dt}\Phi(t,t_0) = A\Phi(t,t_0) \\ \Phi(t,t) = I \end{cases}$$

- But what if we have the controlled system: $\dot{x} = Ax + Bu$
- Claim:

$$x(t) = \Phi(t,t_0)x(t_0) + \int_{t_0}^{t} \Phi(t,\tau)Bu(\tau)d\tau$$

Solving the Controlled Equation

- Claim:

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau$$

$$x(t_0) = \underbrace{\Phi(t_0, t_0)}_{I}x(t_0) + \underbrace{\int_{t_0}^{t_0} \Phi(t_0, \tau)Bu(\tau)d\tau}_{0}$$

$$x(t_0) = x(t_0) \quad \checkmark$$

Solving the Controlled Equation

- Claim:

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau$$

$$\frac{d}{dt}x(t) = A\Phi(t, t_0)x(t_0) + \frac{d}{dt}\int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau$$

$$\frac{d}{dt}\int_{t_0}^{t} f(t, \tau)d\tau = f(t, t) + \int_{t_0}^{t} \frac{d}{dt}f(t, \tau)d\tau$$

$$\Phi(t, t)Bu(t) + \int_{t_0}^{t} A\Phi(t, \tau)Bu(\tau)d\tau$$

Solving the Controlled Equation

- Claim:

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau$$

$$\frac{d}{dt}x(t) = A\Phi(t, t_0)x(t_0) + \frac{d}{dt}\int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau$$

$$\frac{d}{dt}x(t) = A\left(\Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau\right) + Bu(t)$$

$$\frac{d}{dt}x = Ax + Bu \quad \checkmark$$

In Summary

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

$$y(t) = C\Phi(t, t_0)x(t_0) + C\int_{t_0}^{t} \Phi(t, \tau)Bu(\tau)d\tau$$

$$\Phi(t, \tau) = e^{A(t-\tau)}$$

Lecture 3.5 – Stability

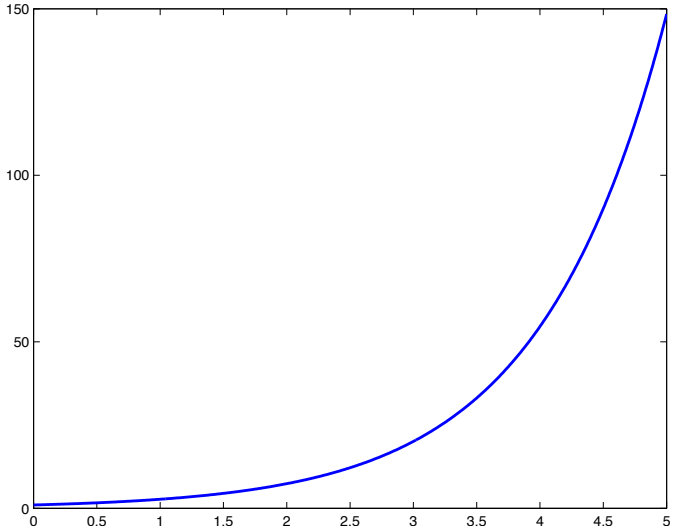
- First order of business is always trying to figure out if the system "blows up" or not
- Recall the control design objectives:
  - Stability
  - Tracking
  - Robustness
  - Other objectives

Scalar Systems

- It is useful to start with scalar systems to get some intuition about what is going on

$$\dot{x} = ax \;\Rightarrow\; x(t) = e^{at}x(0)$$

$a > 0$:

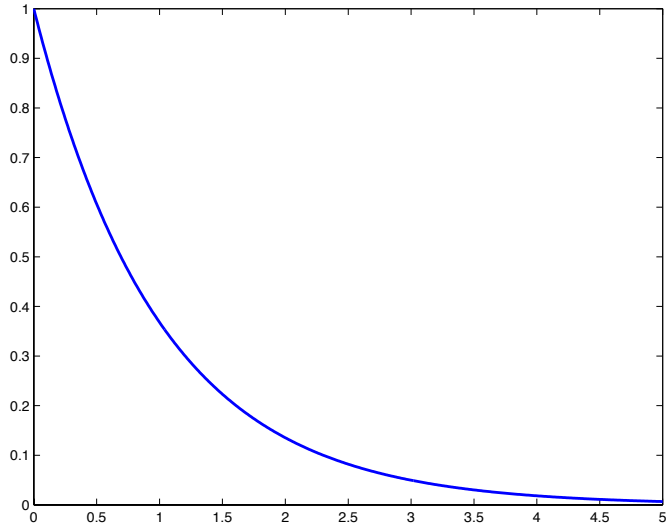


Scalar Systems

- It is useful to start with scalar systems to get some intuition about what is going on

$$\dot{x} = ax \;\Rightarrow\; x(t) = e^{at}x(0)$$

$a < 0$:

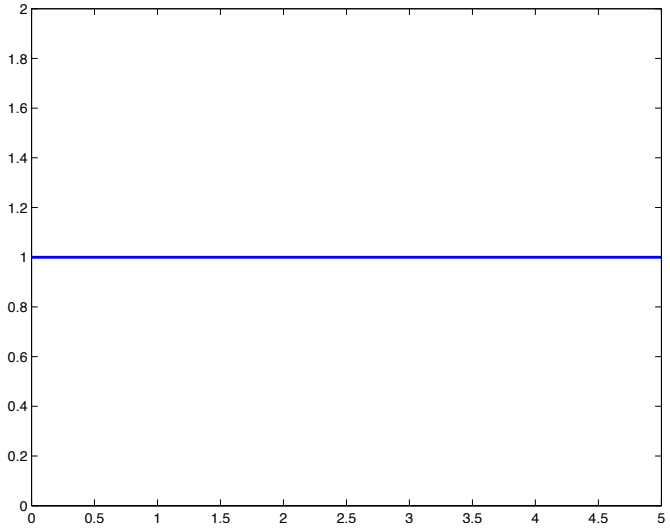


Scalar Systems

- It is useful to start with scalar systems to get some intuition about what is going on

$$\dot{x} = ax \;\Rightarrow\; x(t) = e^{at}x(0)$$

$a = 0$:



Three Cases

- Asymptotically Stable: $x(t) \to 0, \ \forall x(0)$
- Unstable: $\exists x(0) : \ \|x(t)\| \to \infty$
- Critically Stable: in-between (doesn't blow up but doesn't go to zero either)

$$\dot{x} = ax \ \Rightarrow \ x(t) = e^{at}x(0)$$

$$\begin{cases} a > 0 : & \text{unstable} \\ a < 0 : & \text{asymptotically stable} \\ a = 0 : & \text{critically stable} \end{cases}$$

From Scalars to Matrices?

$$\dot{x} = Ax \;\Rightarrow\; x(t) = e^{At}x(0)$$

- We cannot say that A>0, but we can do the next best thing – **eigenvalues**!

$$Av = \lambda v$$

eigenvalue $\in \mathcal{C}$

eigenvector $\in \Re^n$

- The eigenvalues tell us how the matrix A "acts" in different directions (eigenvectors)
- In MATLAB:

```
>> eig(A)
```

Example

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

$$\lambda_1 = 1,\ \lambda_2 = -1,\ v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Stability

$$\dot{x} = Ax \;\Rightarrow\; x(t) = e^{At}x(0)$$

- Asymptotically Stable (if and only if):

$$\mathrm{Re}(\lambda) < 0,\ \forall \lambda \in \mathrm{eig}(A)$$

We will design for this!

- Unstable (if):

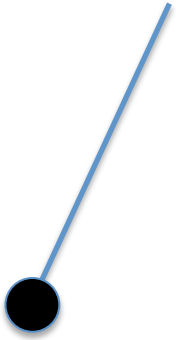
$$\exists \lambda \in \mathrm{eig}(A) :\ \mathrm{Re}(\lambda) > 0$$

- Critically Stable (only if):

$$\mathrm{Re}(\lambda) \leq 0,\ \forall \lambda \in \mathrm{eig}(A)$$

- Critically Stable (if): one eigenvalue is 0 and the rest have negative real part OR two purely imaginary eigenvalues and the rest have negative real part

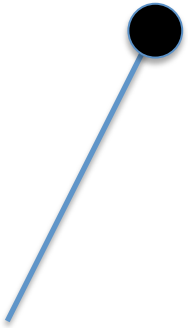
A Tale of Two Pendula



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

$$\lambda_1 = j, \ \lambda_2 = -j$$

critically stable!
oscillates!



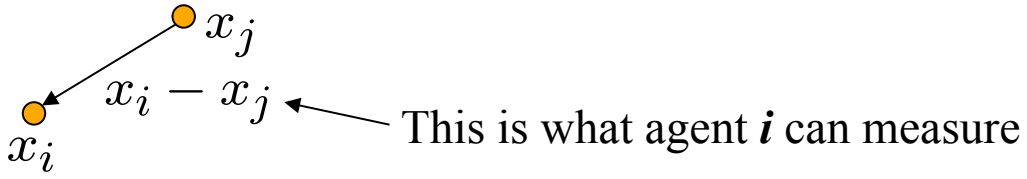
$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\lambda_1 = -1, \ \lambda_2 = 1$$

unstable!

Lecture 3.6 – Swarm Robotics

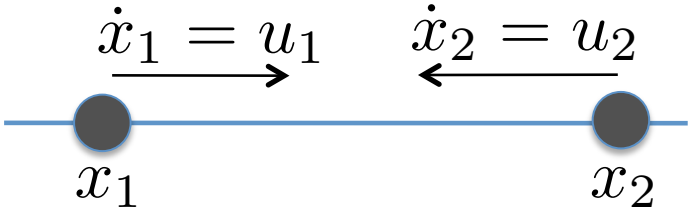
- Let's use the stability results to solve the so-called ***Rendezvous Problem*** in swarm robotics
- The setup:
  - Given a collection of mobile agents who can only measure the relative displacement of their neighbors (no global coordinates)



  - Problem: Have all the agents meet at the same (unspecified) position

The Two-Robot Case

- We have already seen the two-robot case (scalar – does not matter…)



- If the agents simply aim towards each other:

$$\begin{cases} u_1 = x_2 - x_1 \\ u_2 = x_1 - x_2 \end{cases} \qquad \dot{x} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} x$$

The Two-Robot Case

$$A = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \lambda_1 = 0,\ \lambda_2 = -2$$

critically stable

- Fact: If one eigenvalue is 0 and all others have negative real part, then the state will end up in the so-called *null-space* of A

$$\text{null}(A) = \{x \ : \ Ax = 0\}$$

- For this particular A, the null-space is

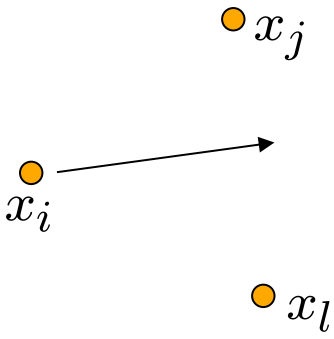
$$\text{null}(A) = \{x \ : \ x = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix},\ \alpha \in \Re\}$$

The Two-Robot Case

- We have that

$$x_1 \to \alpha, \;\; x_2 \to \alpha \;\; \Leftrightarrow \;\; (x_1 - x_2) \to 0$$

- Rendezvous is achieved!
- If there are more than two agents, they should probably aim towards the centroid of their neighbors (or something similar)



$$\dot{x}_i = \sum_{j \in N_i} (x_j - x_i)$$

The Multi-Robot Case

$$\dot{x}_i = \sum_{j \in N_i} (x_j - x_i) \qquad \dot{x} = -Lx$$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

- Fact: If the underlying graph is connected then the graph Laplacian L has one zero eigenvalues and the rest are positive
- Critically Stable!

$$\text{null}(L) = \{x \ : \ x = \begin{bmatrix} \alpha \\ \vdots \\ \alpha \end{bmatrix}\}$$

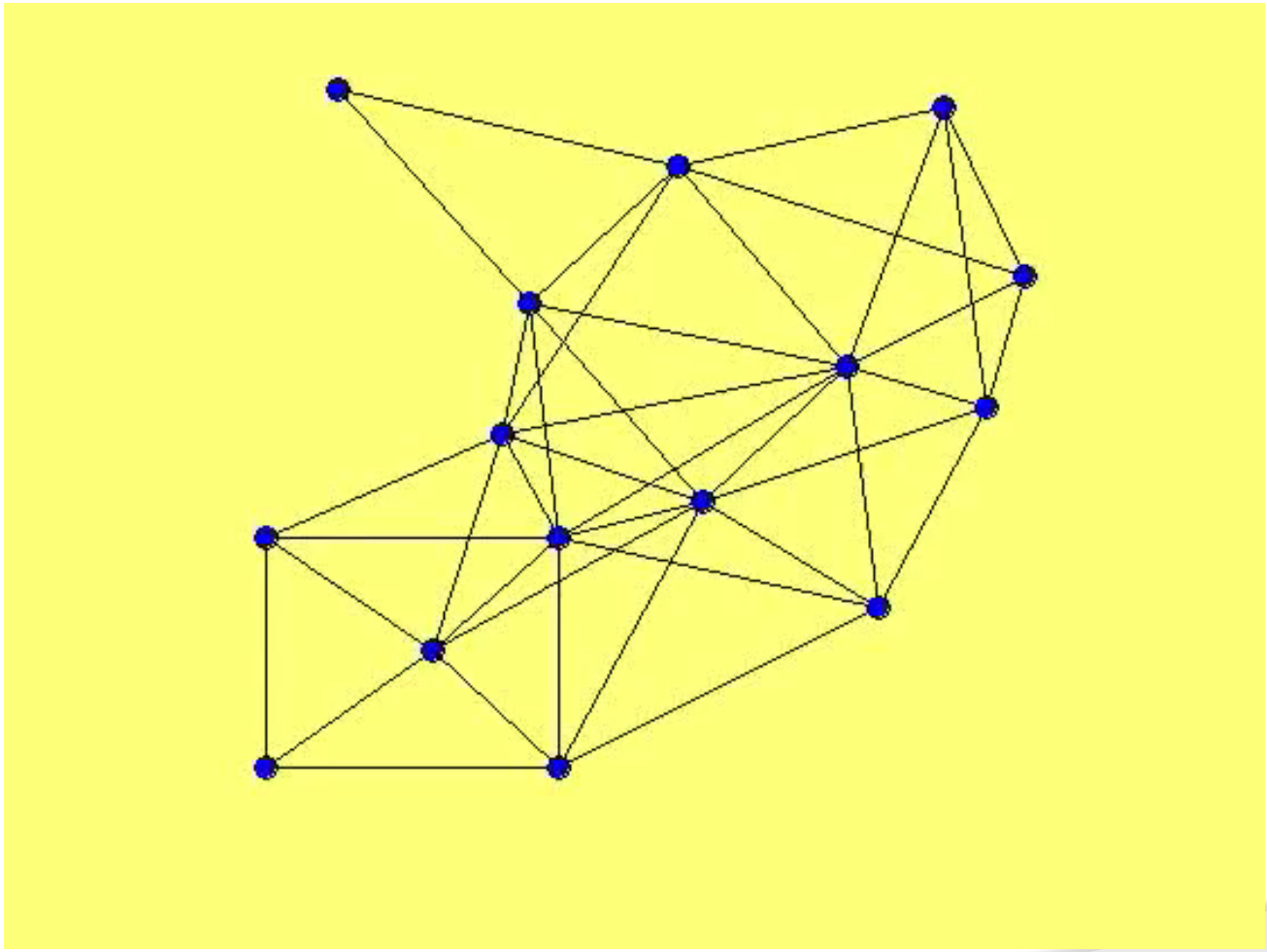
The Multi-Robot Case

$$\dot{x}_i = \sum_{j \in N_i} (x_j - x_i) \qquad \dot{x} = -Lx$$

$$x_i \to \alpha, \ \forall i \ \Leftrightarrow (x_i - x_j) \to 0, \ \forall i, j$$

- Rendezvous is achieved!

Rendezvous



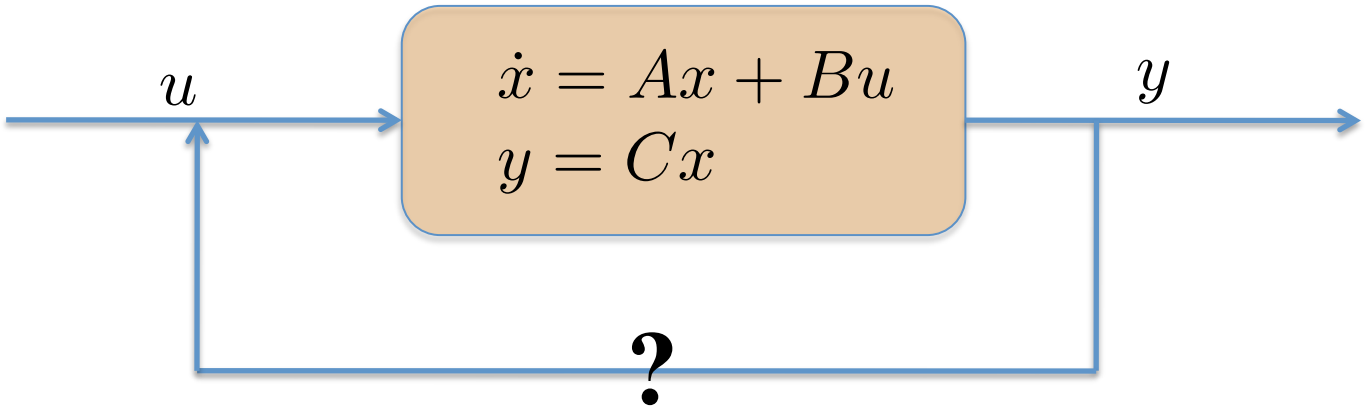
Beyond Rendezvous

And Now For Real

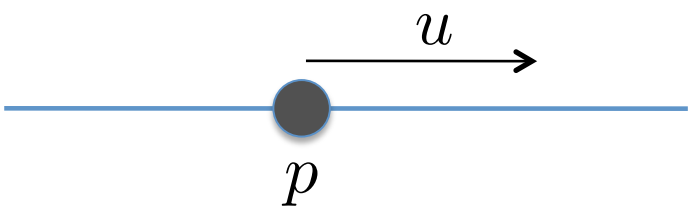
- 2 Kheperas
- Same PID go-to-goal controller as before
- The proposed "high-level" controllers are used to pick a intermediary goal-points
- Robots keep track of position using odometry
- Positions are communicated rather than sensed (later…)

Lecture 3.7 – Output Feedback

- So now we know that the first order of business is to stabilize the system such that all the eigenvalues have negative real part



Back to the World's Simplest Robot



$$\ddot{p} = u$$

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

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

- Idea: Move towards the origin!

$$u > 0 \text{ if } y < 0$$
$$u < 0 \text{ if } y > 0$$

$$u = -y$$

Or, In General

$$u = -Ky = -KCx$$

$$\dot{x} = Ax + Bu = Ax - BKCx = (A - BKC)x$$

- Pick, if possible, K such that

$$\mathrm{Re}(\lambda) < 0 \ \forall \lambda \in \mathrm{eig}(A - BKC)$$

Back to the Robot

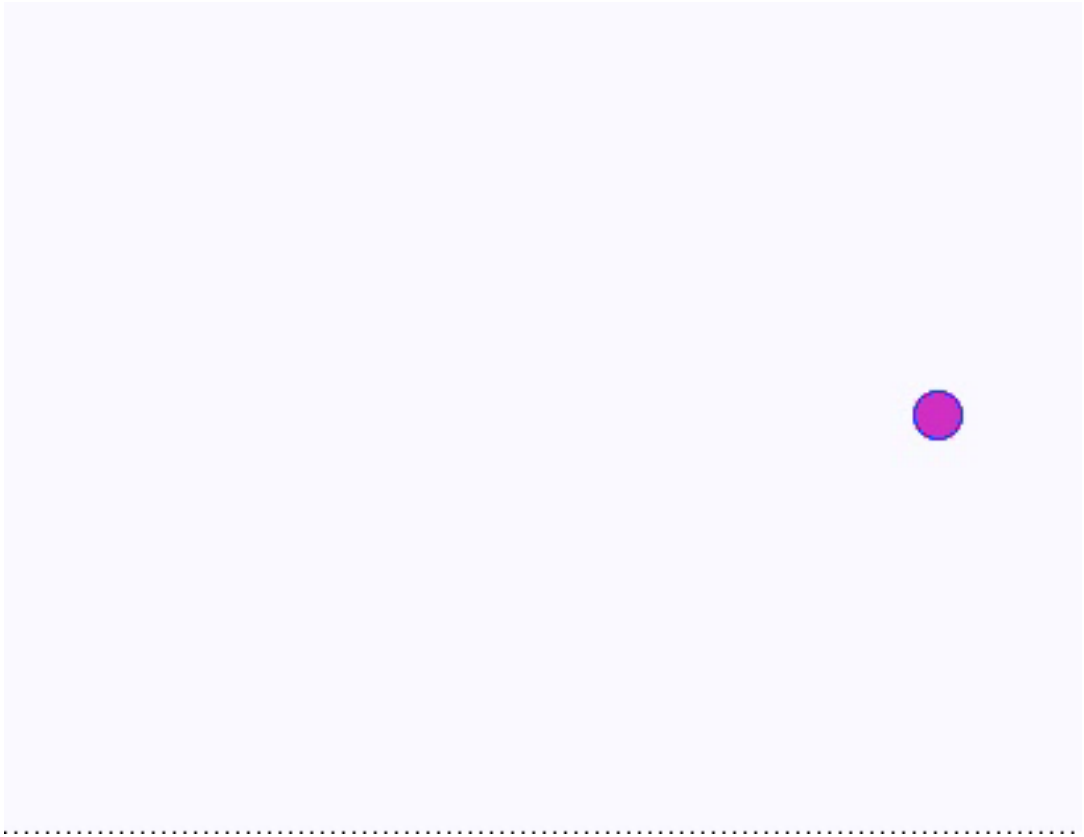
$$\dot{x} = \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} 1 \begin{bmatrix} 1 & 0 \end{bmatrix} \right) x$$

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

$$\text{eig}(A - BKC) = \pm j$$

Critically stable!

Back to the Robot

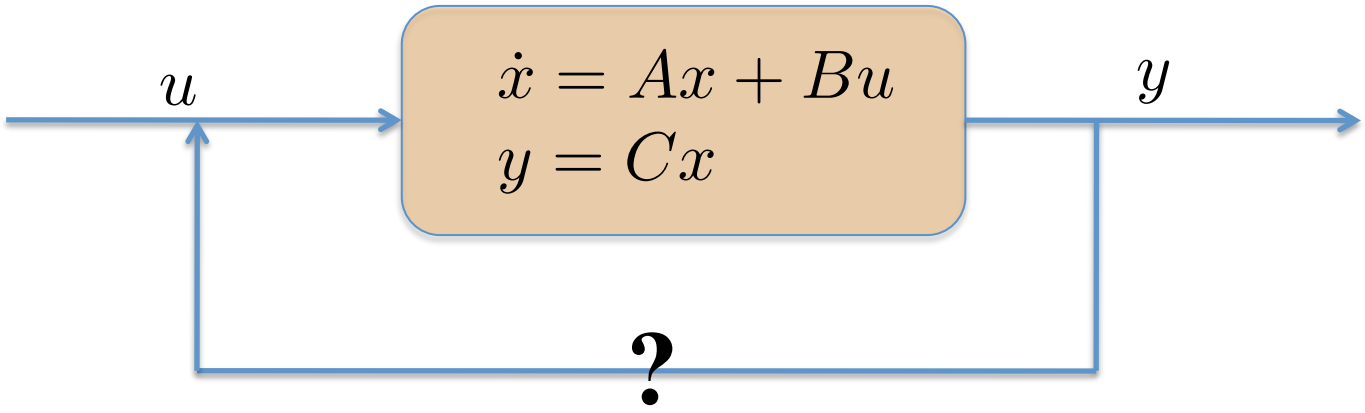


What's The Problem?

- The problem is that we do not take the velocity into account!
- We need to use the full state information in order to stabilize this system!
- Problem 1 (How to do that?)
- Problem 2 (But we do not know x, we know y?)
- Next Module! (and some next lecture…)

Lecture 3.8 – State Feedback

- Need to stabilize the system such that all the eigenvalues have negative real part – Using state feedback!



Closing the Loop

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

$$u = -Kx$$

closed-loop dynamics

$$\dot{x} = Ax + Bu = Ax - BKx = (A - BK)x$$

- Pick, if possible, K such that the closed-loop system is stabilized, i.e.,

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

- Module 4!

Back to the Robot

$$u \in \Re,\ x \in \Re^2,\ K : 1 \times 2$$

$$K = \begin{bmatrix} k_1 & k_2 \end{bmatrix}$$

$$\dot{x} = \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} \right) x$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} x$$

Picking the Gains

- In the next module, we will pick gains in a systematic manner, but for now, let's try

$$k_1 = k_2 = 1$$

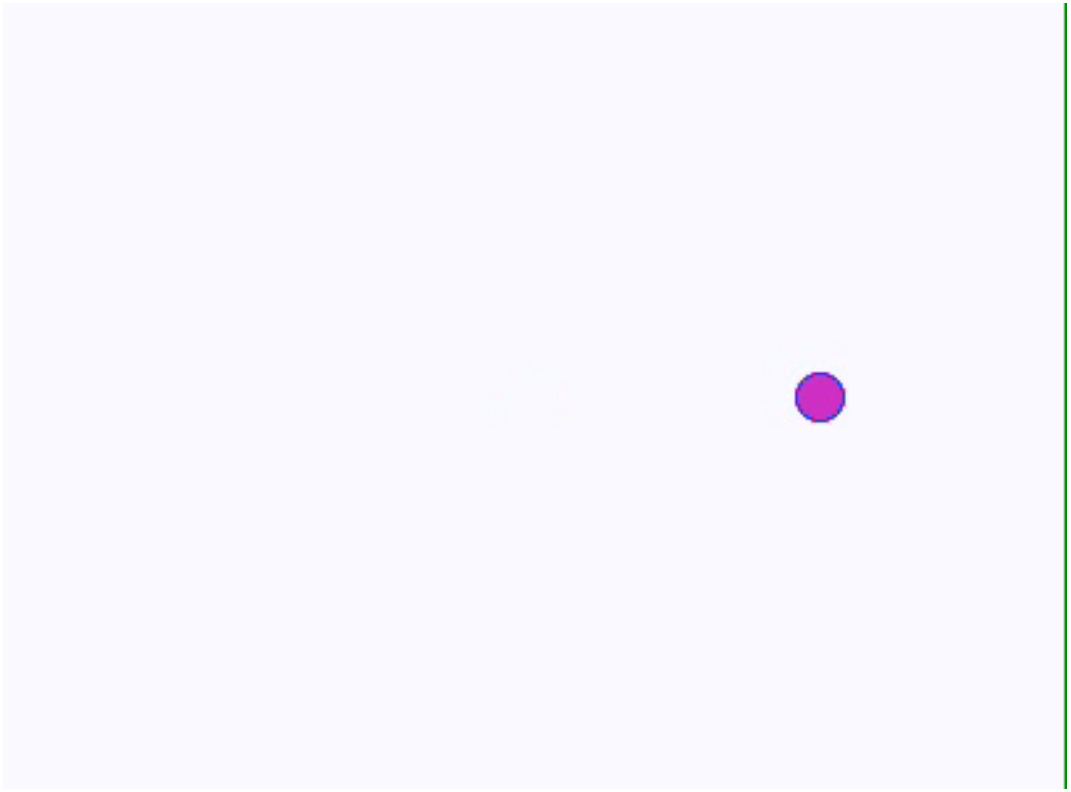
$$A - BK = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}$$

$$\text{eig}(A - BK) = -0.5 \pm 0.866j$$

Asymptotically stable!

Damped oscillations

Attempt 1



Another Attempt

$$k_1 = 0.1, \;\; k_2 = 1$$

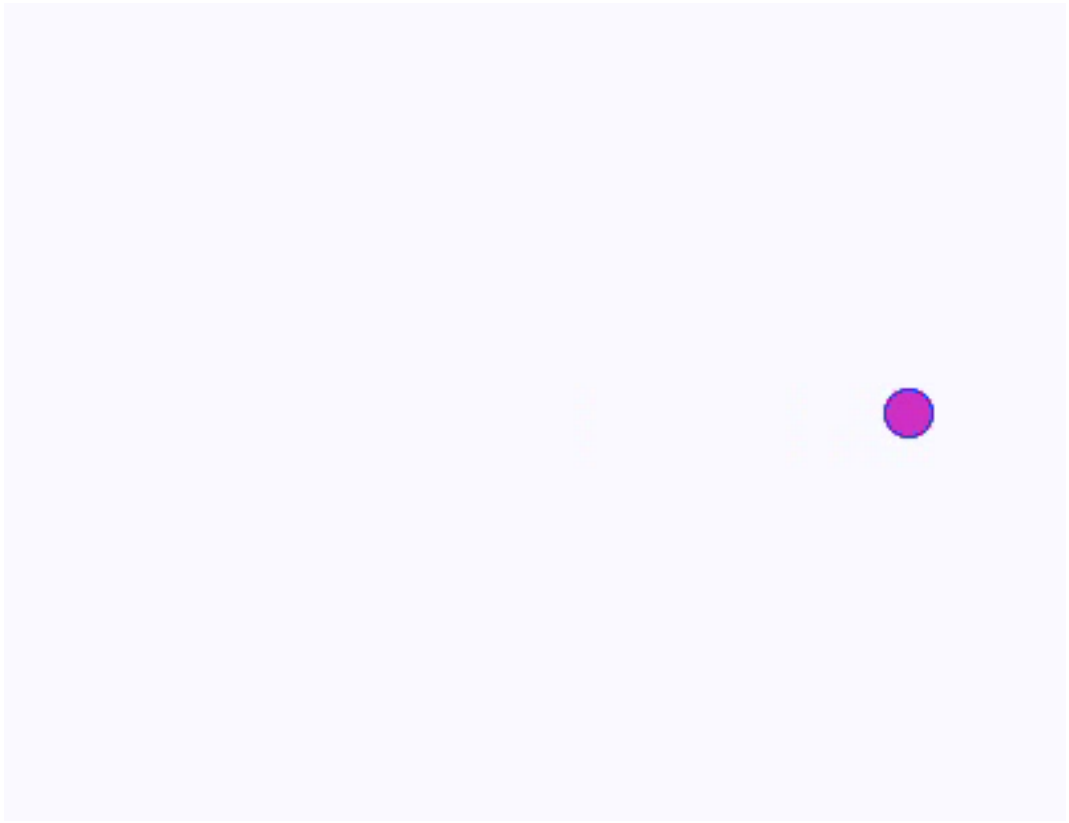
$$A - BK = \begin{bmatrix} 0 & 1 \\ -0.1 & -1 \end{bmatrix}$$

$$\text{eig}(A - BK) = -0.1127, -0.8873$$

Asymptotically stable!

No oscillations

Attempt 2



Eigenvalues Matter

- It is clear that some eigenvalues are better than others. Some cause oscillations, some make the system respond too slowly, and so forth…
- In the next module we will see how to select eigenvalues and how to pick control laws based on the output rather than the state.