three ideas inspired by biology for how to improve robotics

(the focus of this course)

1. adaptation through evolution and learning

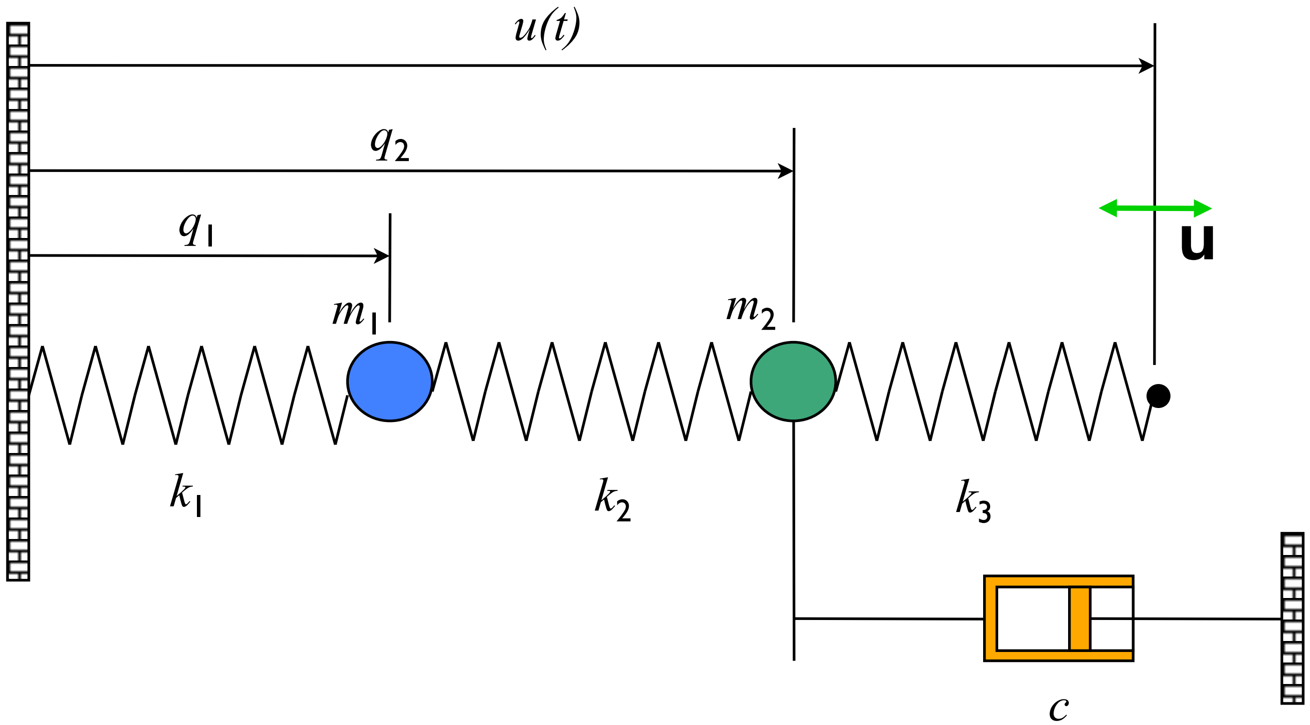
😀 fundamental engineering processes used by biology
☹️ "curse of dimensionality"

2. mechanical intelligence
   - the use of mechanics to reduce or eliminate the need for feedback control
3. parsimony
   - simple and efficient solutions

😀 "shortcut": look directly to biology for inspiration, combine with engineering knowledge

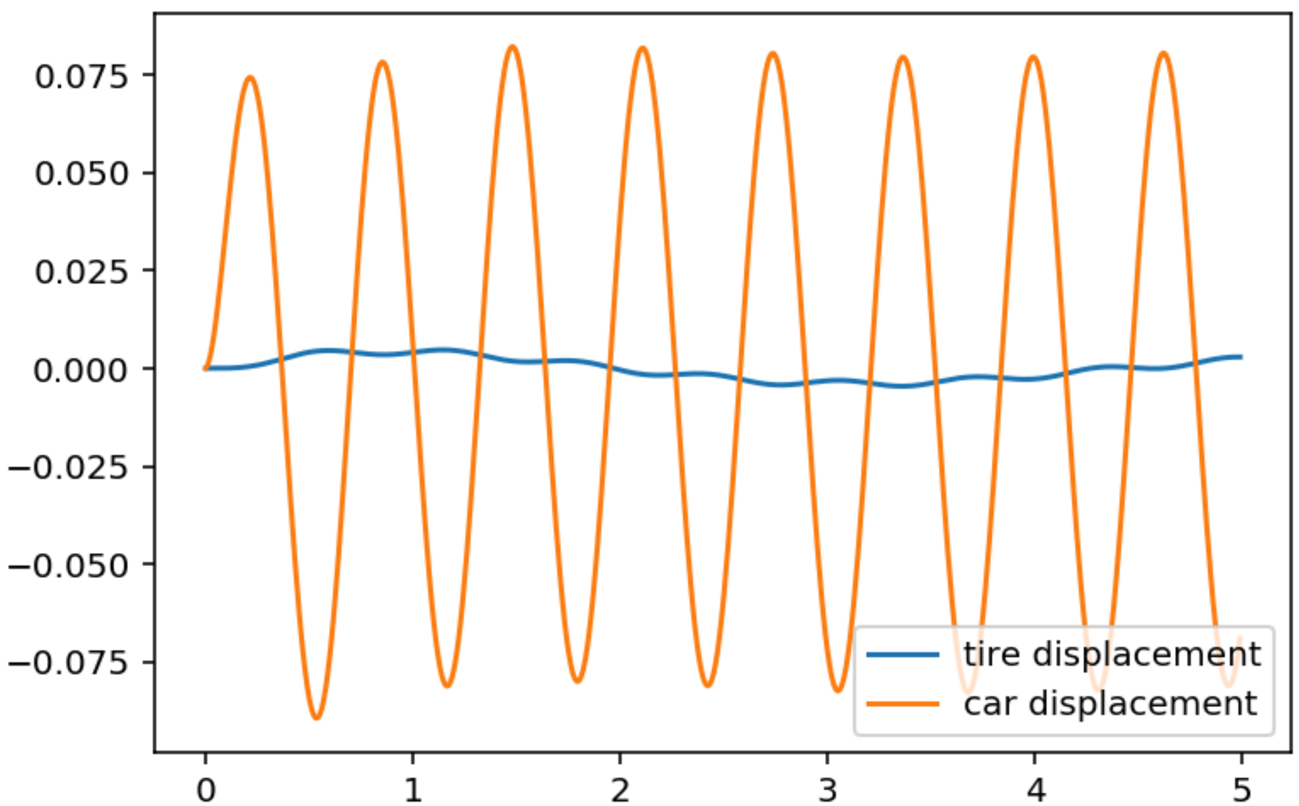
⇒ the approach emphasized in ME586 homework and projects

Simulating a state-space system



basic task: repeatedly calculate state update:

$$\boldsymbol{q}_{t+\Delta T} = \boldsymbol{q}_t + \Delta T \dot{\boldsymbol{q}}_t = \boldsymbol{q}_t + \Delta T \boldsymbol{f}(\boldsymbol{q})$$



Python simulation

```
import numpy as np
import matplotlib.pyplot as plt

k1 = k2 = k3 = m1 = c = 1
m2 = 0.1
dt = 0.01

time = np.arange(0, 5, dt)
q_data = np.zeros((len(time), 4))
q = np.array((0, 0, 0, 0))      ← initial condition
def f(q, u):                    ← dynamics function
     return np.array((
          q[2],
          q[3],
          -(k1+k2)/m1*q[0] + k2/m1*q[1],
          k2/m2*q[0] - (k2+k3)/
              m2*q[1] - c/m2*q[3] + k3/m2*u))

for idx, t in enumerate(time):
    u = np.cos(10*t)
    q = q + dt * f(q, u)         ← update step
    q_data[idx,:] = q            ← store result
plt.plot(time, q_data[:,0:2])
plt.legend(('car displacement (q1)',
            'tire displacement (q2)'))
```

general form of differential equations

State space form

$$\frac{dx}{dt} = f(x,u) \qquad y = h(x,u)$$

General form

$$\frac{dx}{dt} = Ax + Bu \qquad y = Cx + Du$$

Linear system

$$x \in \mathbb{R}^n,\ u \in \mathbb{R}^p \qquad y \in \mathbb{R}^q$$

• x = state; *n*th order

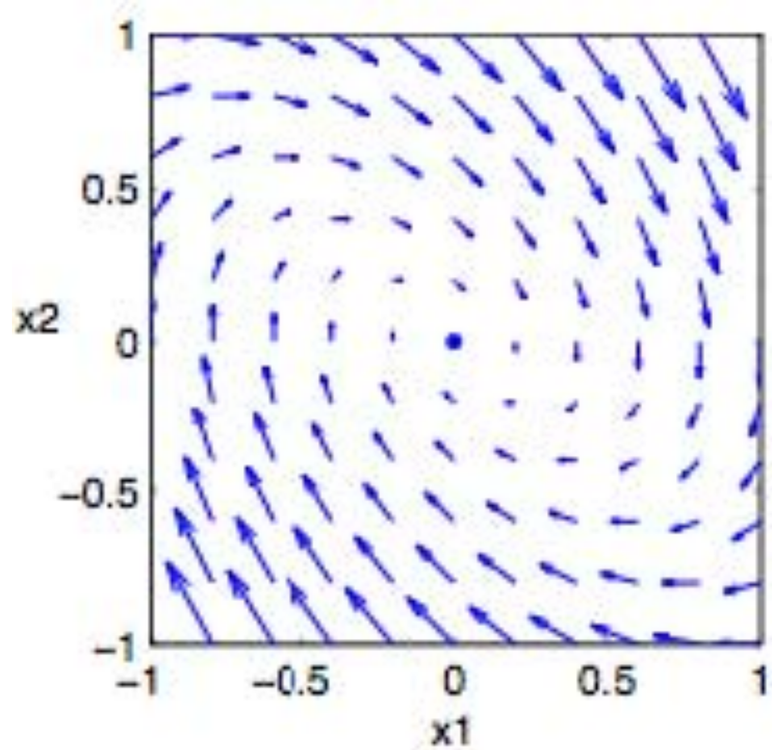
phase plots show 2D behavior

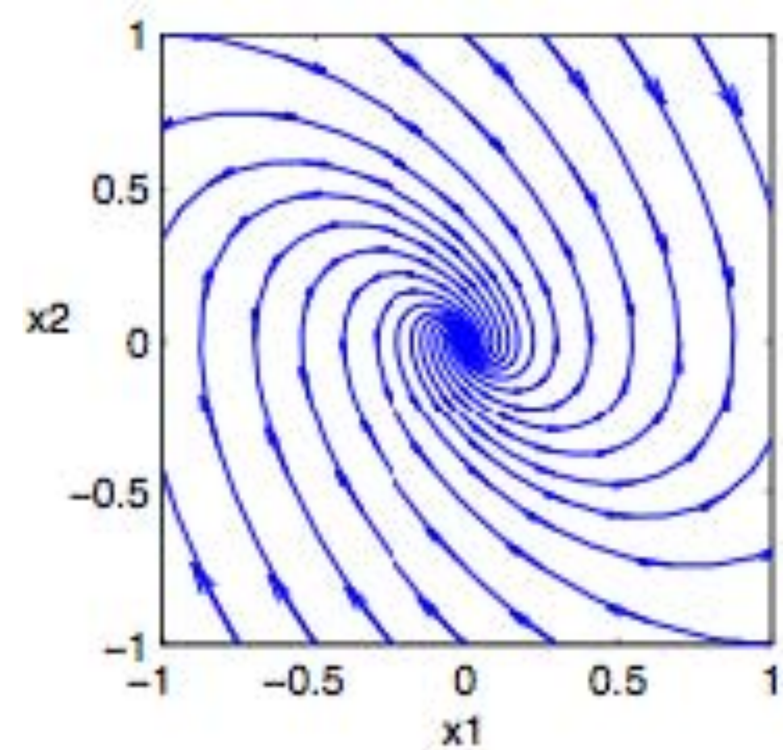
Phase plane plots show 2D dynamics as *vector fields* & *stream functions*

- $\dot{x} = f(x, u(x)) = F(x)$
- Plot *F*(*x*) as a vector on the plane; stream lines follow the flow of the arrows

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 - x_2 \end{bmatrix}$$

python: use 'streamplot' function in Matplotlib





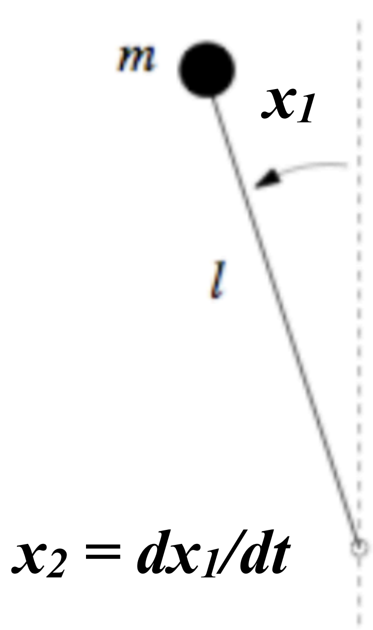
equilibrium points

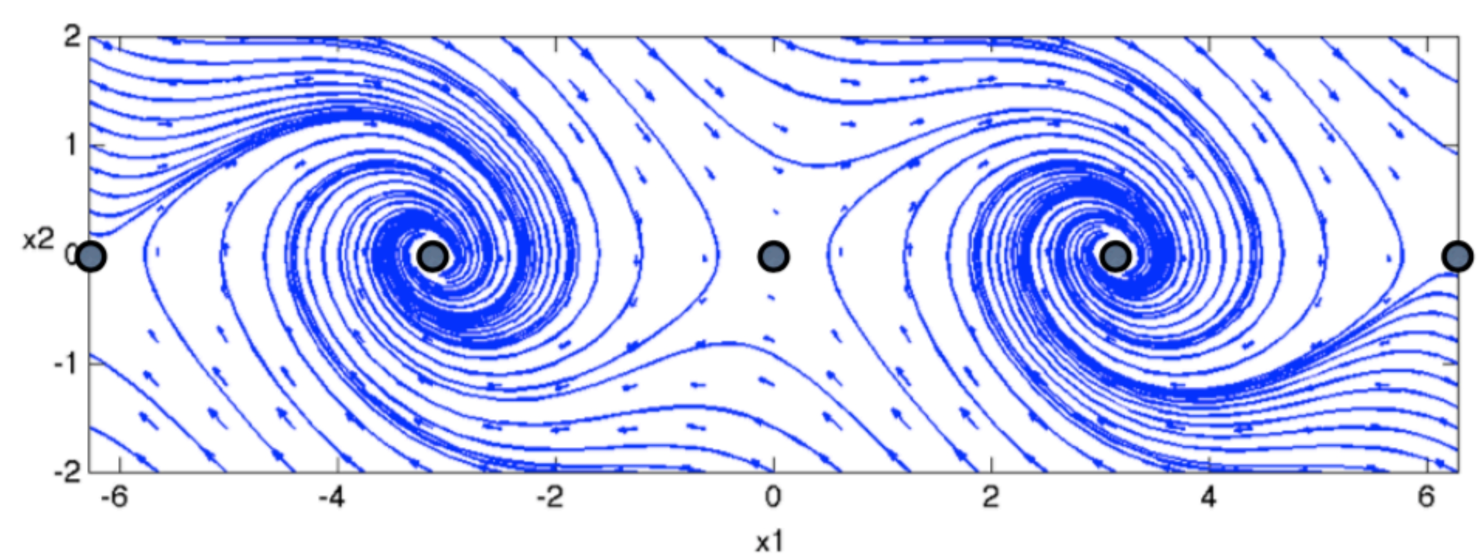
Equilibrium points represent stationary conditions for the dynamics

The *equilibria* of the system $\dot{x} = f(x)$ are the points x_e such that $f(x_e) = 0$.

Example:

$$\frac{dx}{dt} = \begin{bmatrix} x_2 \\ \sin x_1 - \gamma x_2 \end{bmatrix} \quad \Rightarrow \quad x_e = \begin{bmatrix} \pm n\pi \\ 0 \end{bmatrix}$$



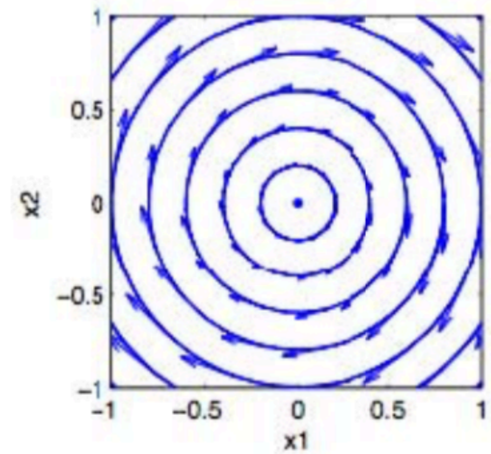


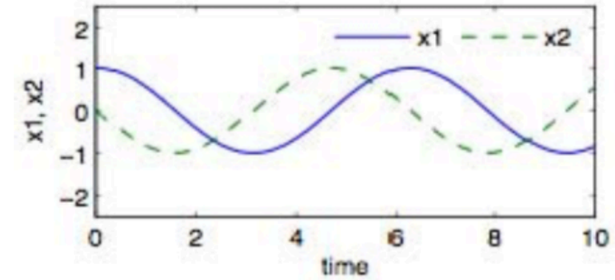
stability of equilibrium points

An equilibrium point is:

Stable if initial conditions that start near the equilibrium point, stay near

- Also called “stable in the sense of Lyapunov

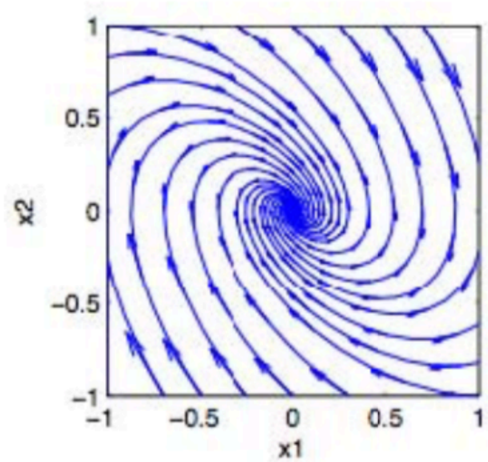


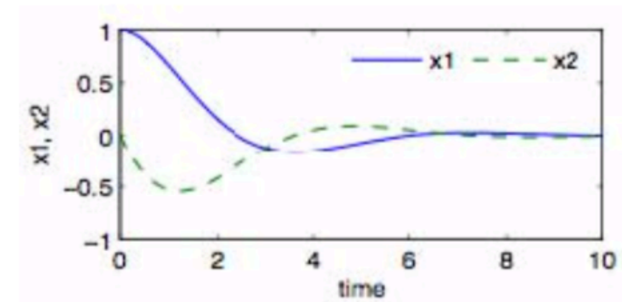


“stable” but not asymptotically stable

Asymptotically stable if all nearby initial conditions converge to the equilibrium point

- Stable + converging

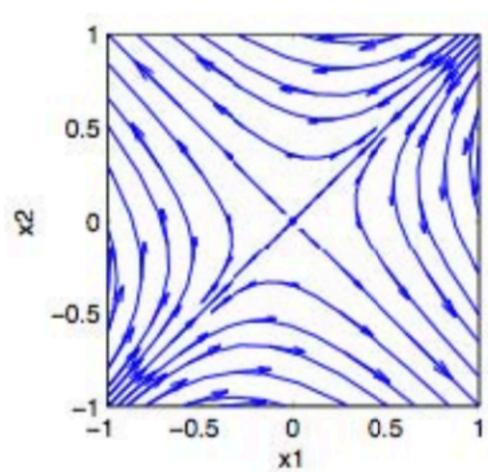


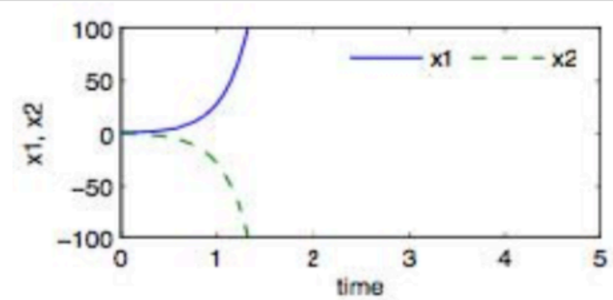


asymptotically stable

Unstable if some initial conditions diverge from the equilibrium point

- May still be some initial conditions that converge



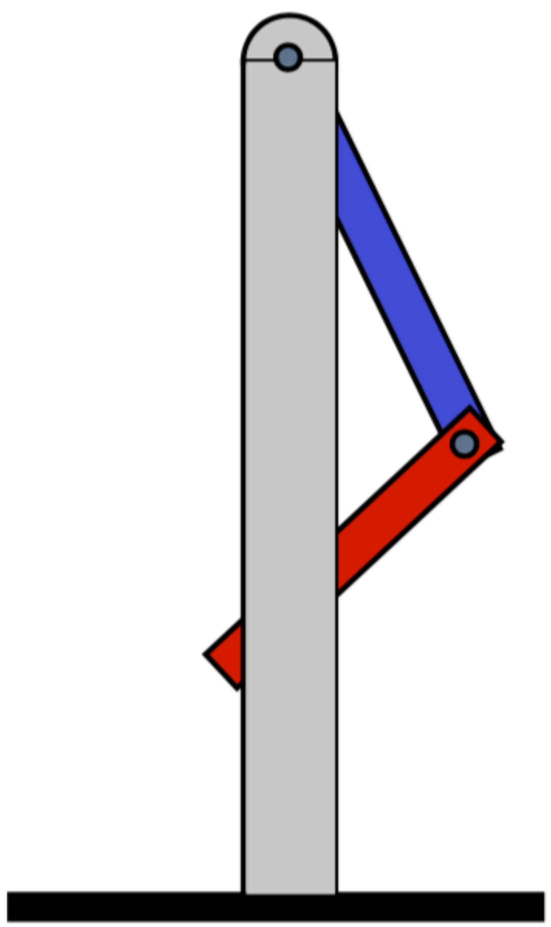


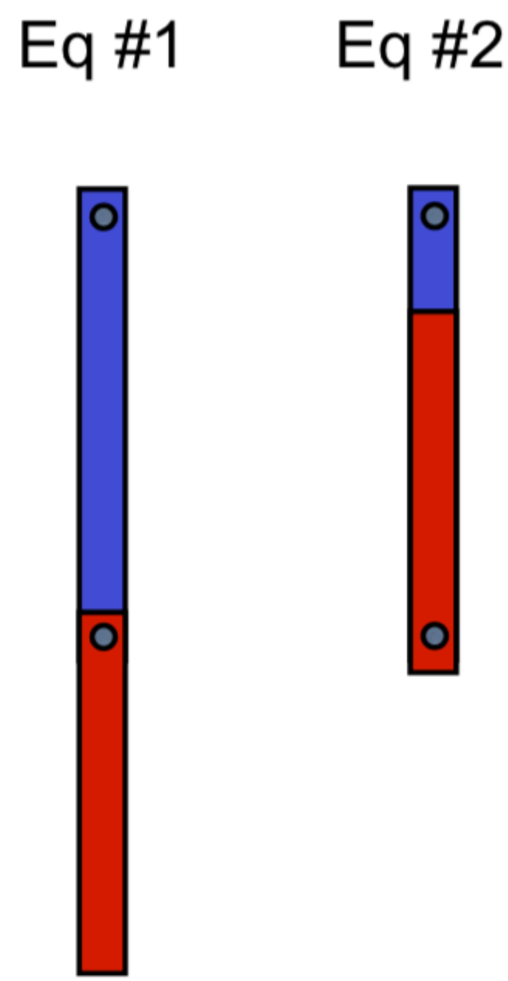
unstable

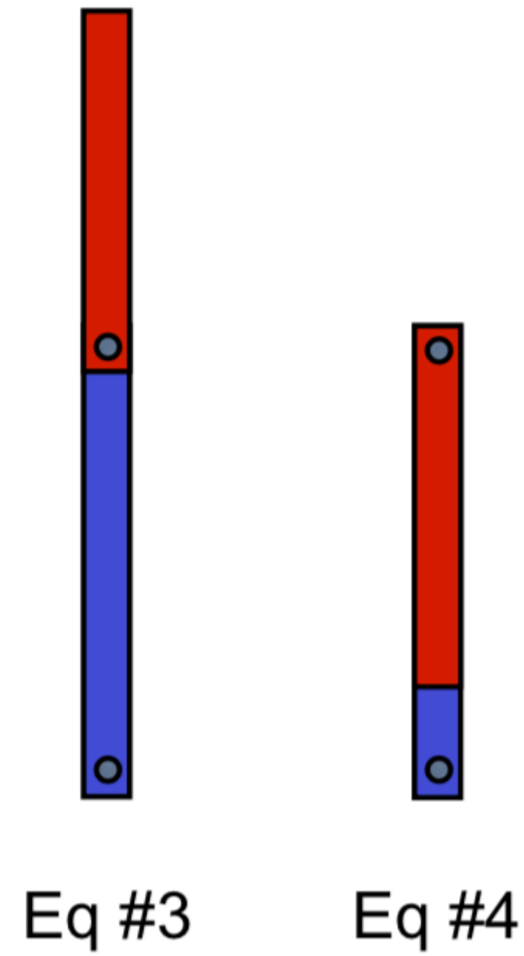
Example #1: Double Inverted Pendulum

Two series coupled pendula

- States: pendulum angles (2), velocities (2)
- Dynamics: $F = ma$ (balance of forces)
- Dynamics are very nonlinear







Stability of equilibria

- Eq #1 is stable
- Eq #3 is unstable
- Eq #2 and #4 are unstable, but with some stable "modes"

Stability of linear systems $\dot{x} = Ax$

- Theorem: linear system is asymptotically stable if and only if all eigenvalues λ of A have negative real part.

Local stability of nonlinear systems $\dot{x} = F(x)$

Asymptotic stability of the linearization implies *local* asymptotic stability of equilibrium point

- Linearization around equilibrium point captures "tangent" dynamics

$$\dot{x} = \underbrace{F(x_e)}_{0} + \left.\frac{\partial F}{\partial x}\right|_{x_e} (x - x_e) + \text{higher order terms} \xrightarrow{\text{approx}} \begin{array}{l} z = x - x_e \\ \dot{z} = Az \end{array}$$

- linearization is *stable* ⟹ nonlinear system *locally stable*
- linearization is *unstable* ⟹ nonlinear system *locally unstable*

- "degenerate case": if linearization is *stable* but not *asymptotically stable* ⟹ cannot tell whether nonlinear system is stable or not!

$$\dot{x} = \pm x^3 \xrightarrow{\text{linearize}} \dot{x} = 0$$

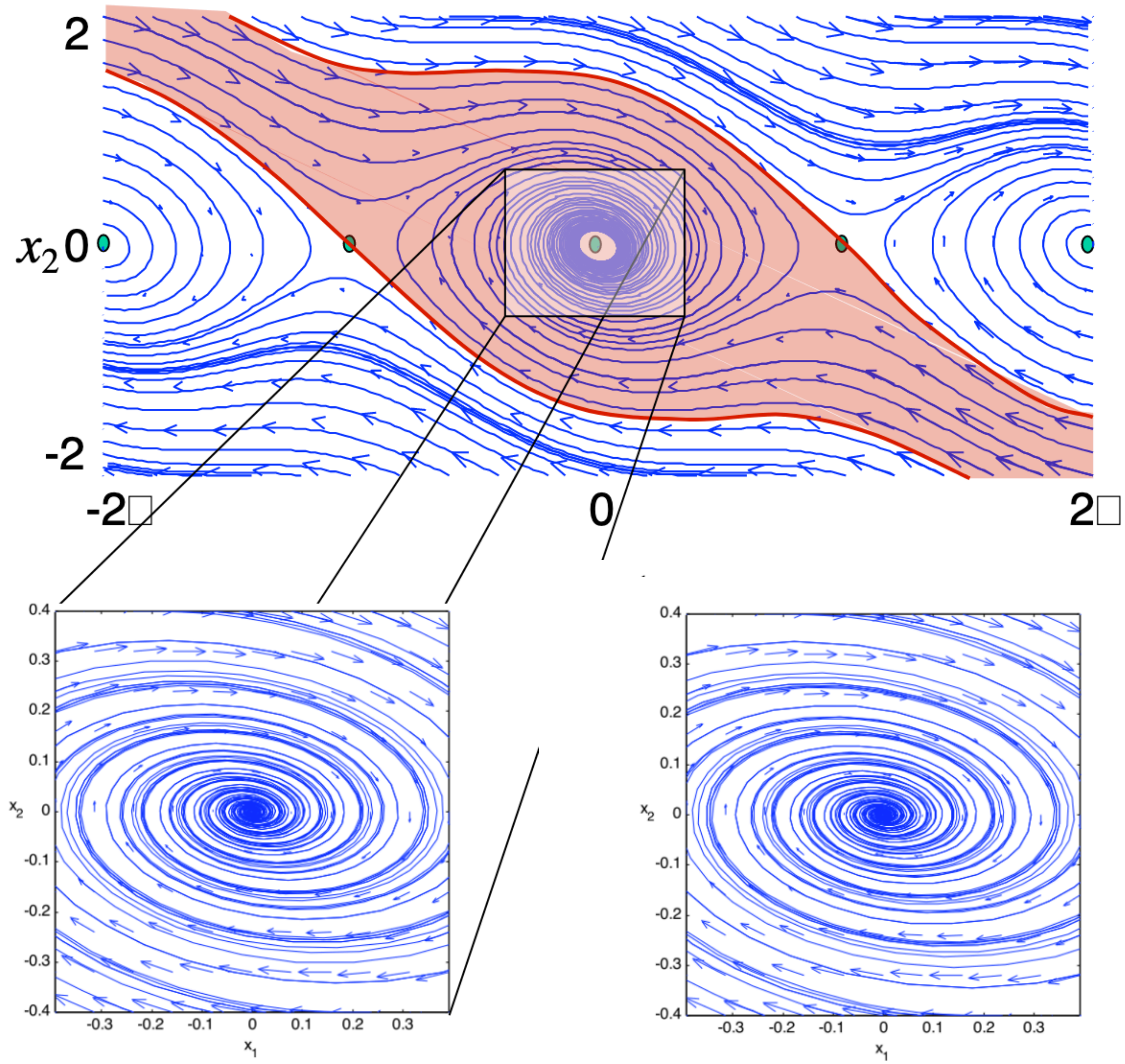
- linearization is stable (but not asy stable)
- nonlinear system can be asy stable or unstable

Local linear approximation is valuable for control design:

- if dynamics are well-approximated by linearization near an equilibrium point, controller can *ensure* stability there (!)
- controller task: make the linearization stable

Linearization about an equilibrium point

$$\dot{x} = f(x,u) \qquad \dot{z} = Az + Bv$$
$$y = h(x,u) \quad \longrightarrow \quad w = Cz + Dv$$



full nonlinear model **linear model (honest!)**

to "linearize" around $x = x_e$:

1. **find x_e, u_e such that $f = 0$**
2. **define** $y_e = h(x_e, u_e)$

   $z = x - x_e \quad v = u - u_e \quad w = y - y_e$

3. **then**

$$A = \left.\frac{\partial f}{\partial x}\right|_{(x_e,u_e)} \qquad B = \left.\frac{\partial f}{\partial u}\right|_{(x_e,u_e)}$$

$$C = \left.\frac{\partial h}{\partial x}\right|_{(x_e,u_e)} \qquad D = \left.\frac{\partial h}{\partial u}\right|_{(x_e,u_e)}$$

Remarks

- In examples, this is often equivalent to small angle approximations, etc
- Only works *near* to equilibrium point
- use linearization to design controller

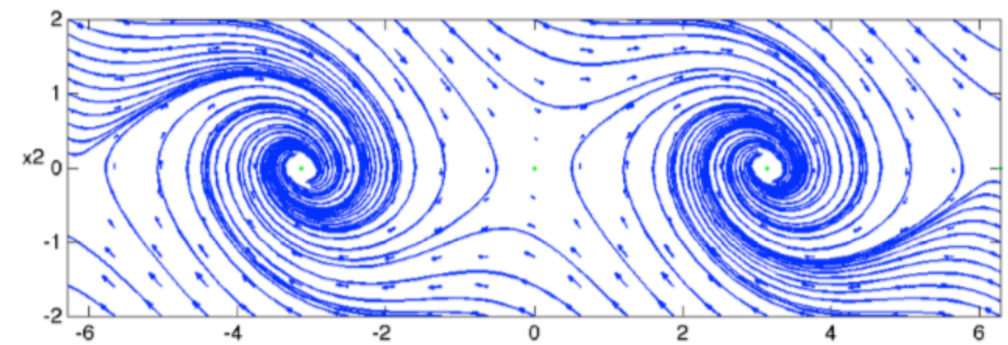
big idea: if combined linearized system + controller is stable $\Longrightarrow$ nonlinear system (incl control) is stable nearby

Jacobian linearization matrix

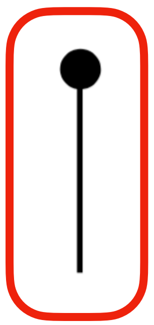
$$A = \left.\frac{\partial f}{\partial x}\right|_{(x_e, u_e)} = \left.\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}\right|_{(x_e,\, u_e)}$$

Example: Stability Analysis of Inverted Pendulum

System dynamics



$$\frac{dx}{dt} = \begin{bmatrix} x_2 \\ \sin x_1 - \gamma x_2 \end{bmatrix},$$

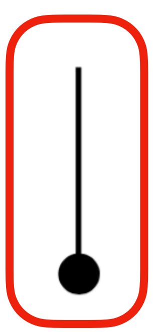


Upward equilibrium:

$$\theta = x_1 \ll 1 \quad \Longrightarrow \quad \sin x_1 \approx x_1$$

$$\frac{dx}{dt} = \begin{bmatrix} x_2 \\ x_1 - \gamma x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -\gamma \end{bmatrix} x$$

- Eigenvalues: $-\frac{1}{2}\gamma \pm \frac{1}{2}\sqrt{4+\gamma^2}$ **for** $\gamma = 0.1,\ \lambda \cong (0.95, -1.05) \Rightarrow$ **unstable**



Downward equilibrium:

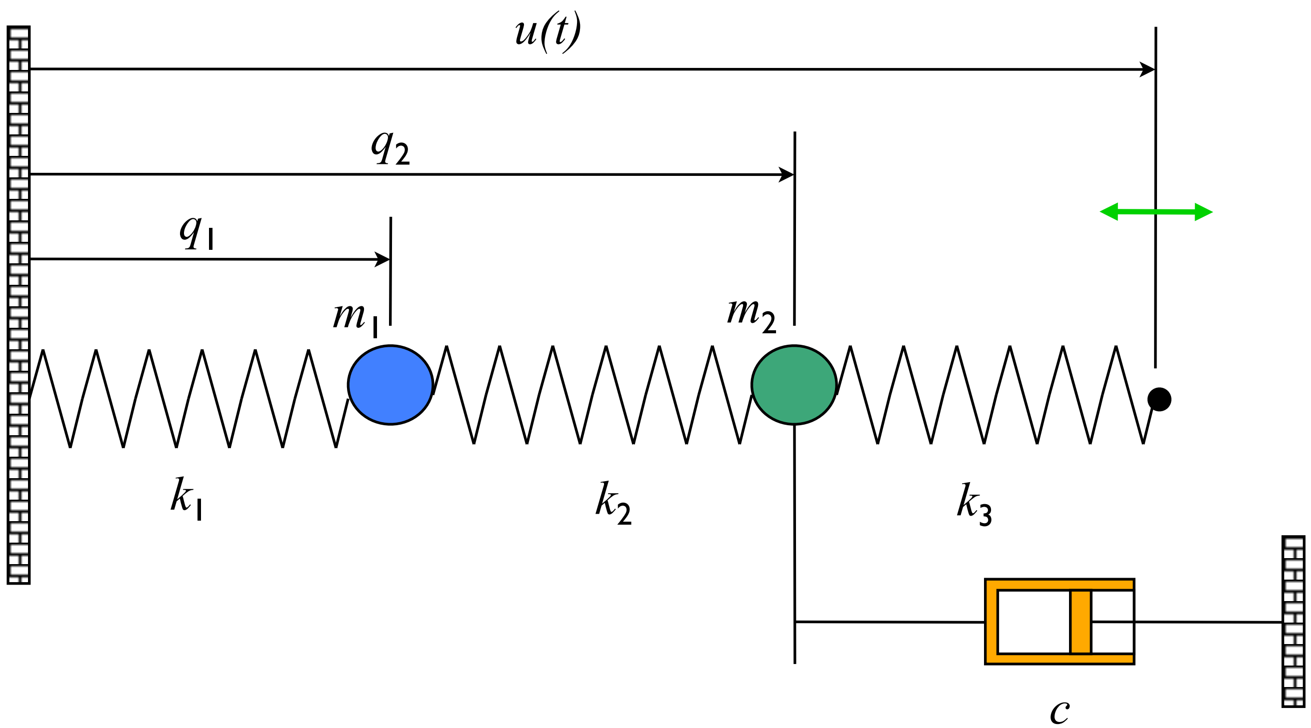
- Linearize around $x_1 = \pi + z_1$: $\sin(\pi + z_1) = -\sin z_1 \approx -z_1$
- Eigenvalues:

$$z_1 = x_1 - \pi$$
$$z_2 = x_2$$

$$\longrightarrow \quad \frac{dz}{dt} = \begin{bmatrix} z_2 \\ -z_1 - \gamma\ z_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -\gamma \end{bmatrix} z$$

$$-\frac{1}{2}\gamma \pm \frac{1}{2}\sqrt{-4+\gamma^2}$$ **for** $\gamma = 0.1,\ \lambda \cong (-0.05+i, -0.05-i) \Rightarrow$ **stable**

example 2: matrix representation of a linear system



Model: rigid body physics

- Sum of forces = mass * acceleration
- Hooke's law: $F = k(x - x_{rest})$
- Viscous friction: $F = c\, v$

$$m_1\ddot{q}_1 = k_2(q_2 - q_1) - k_1 q_1$$
$$m_2\ddot{q}_2 = k_3(u - q_2) - k_2(q_2 - q_1) - c\dot{q}_2$$

$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \frac{k_2}{m}(q_2 - q_1) - \frac{k_1}{m}q_1 \\ \frac{k_3}{m}(u - q_2) - \frac{k_2}{m}(q_2 - q_1) - \frac{c}{m}\dot{q} \end{bmatrix}$$

$$y = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

"State space form"

Matrix representation:

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

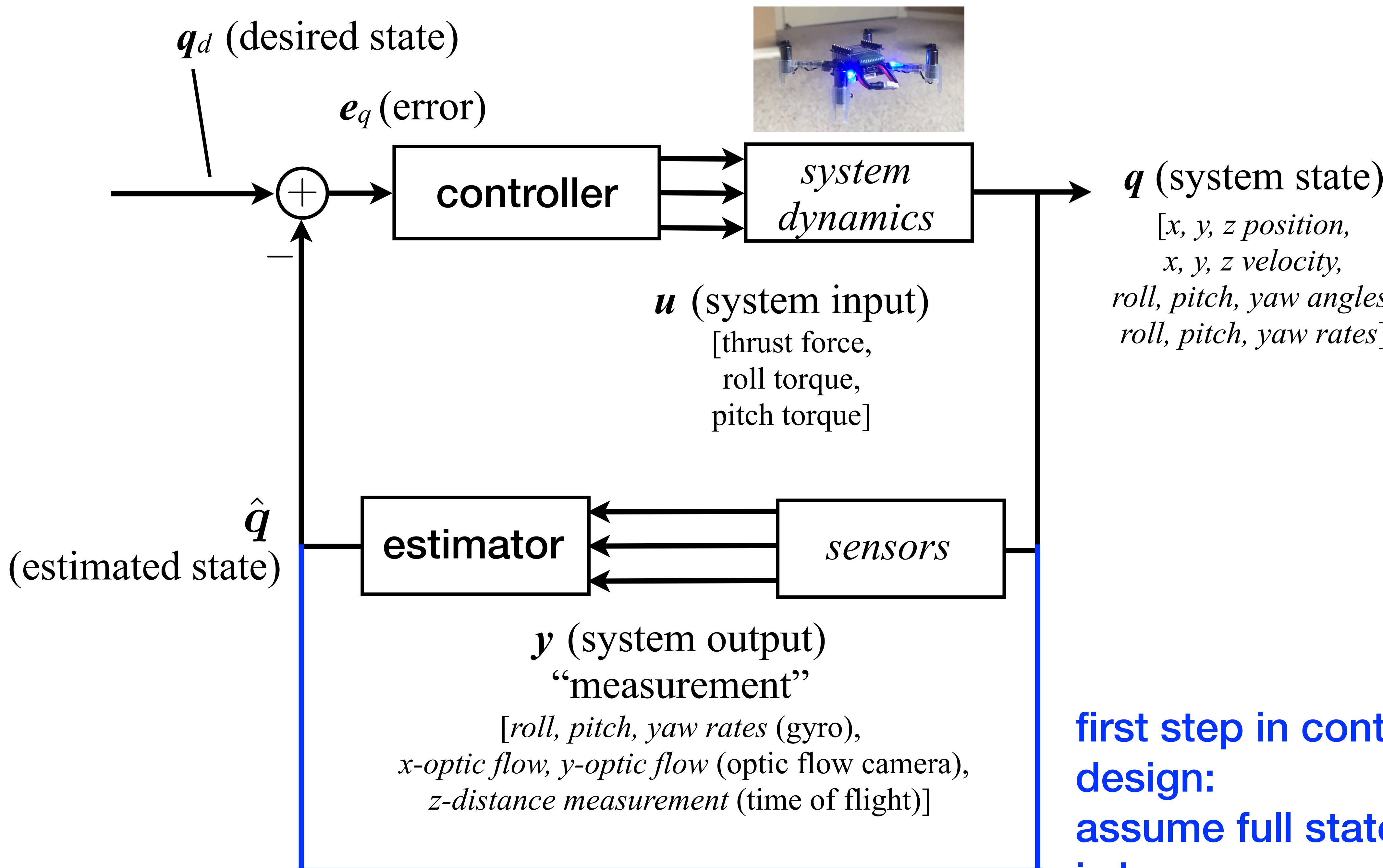
$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1 + k_2}{m} & \frac{k_2}{m} & 0 & 0 \\ -\frac{k_2}{m} & -\frac{k_2 + k_3}{m} & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{k_3}{m} \end{bmatrix} u$$

$$y = [1 \quad 1 \quad 0 \quad 0]x = Cx$$

Upcoming/today

- Nonlinear dynamics and stability
- state feedback and “reachability/controllability”
- control and simulation of Newton-Euler Equations of motion
- LQR control

the control task



first step in control design:
assume full state is known

State Space Control Design Concepts

System description: single input, single output system (MIMO also OK)

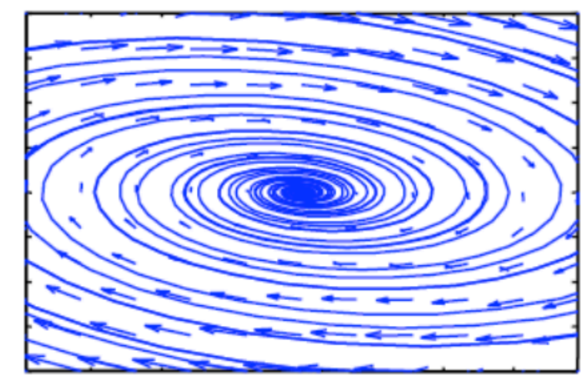
$$\dot{x} = f(x,u) \qquad x \in \mathbb{R}^n,\ x(0) \text{ given}$$
$$y = h(x) \qquad u \in \mathbb{R},\ y \in \mathbb{R}$$

Stability: stabilize the system around an equilibrium point

- Given equilibrium point $x_e \in \mathbb{R}^n$, find control "law" $u = \alpha(x)$ such that

$$\lim_{t\to\infty} x(t) = x_e \text{ for all } x(0) \in \mathbb{R}^n$$

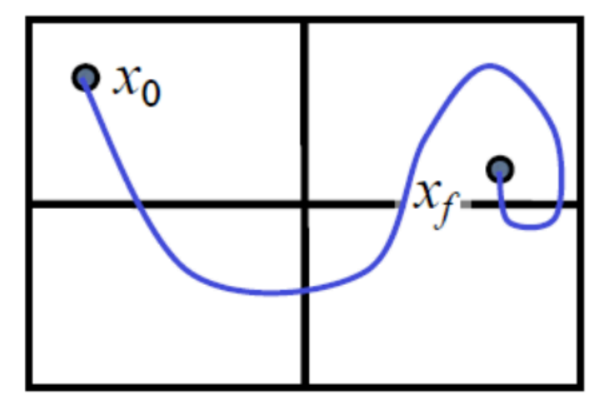
- Often choose x_e so that $y_e = h(x_e)$ has desired value r (constant)



Reachability: steer the system between two points

- Given $x_o, x_f \in \mathbb{R}^n$, find an input $u(t)$ such that

$$\dot{x} = f(x, u(t)) \text{ takes } x(t_0) = x_0 \to x(T) = x_f$$



Tests for Reachability

$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n,\ x(0) \text{ given}$$
$$y = Cx \qquad u \in \mathbb{R},\ y \in \mathbb{R}$$

$$x(T) = e^{AT}x_0 + \int_{\tau=0}^{T} e^{A(T-\tau)}Bu(\tau)d\tau$$

Thm A linear system is reachable if and only if the $n \times n$ *reachability matrix*

$$\begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

is full rank.

Note: also called "controllability" matrix

Remarks

- *Very* simple test : `control.ctrb(A,B)` and check rank with `numpy.linalg.matrix_rank()`
- If this test is satisfied, we say "the pair (A,B) is reachable"

State space controller design for linear systems

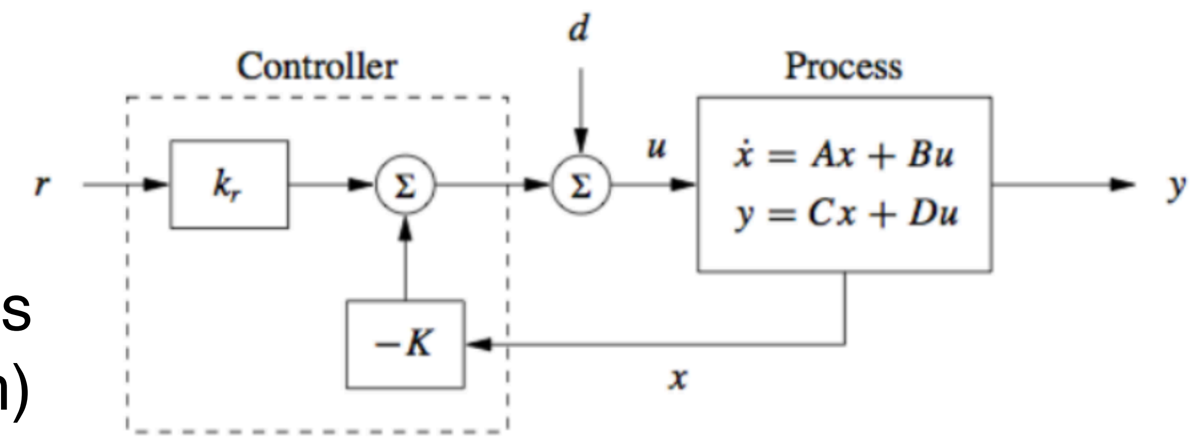
$$\dot{x} = Ax + Bu \quad x \in \mathbb{R}^n,\ x(0) \text{ given}$$
$$y = Cx \quad u \in \mathbb{R},\ y \in \mathbb{R}$$

$$x(T) = e^{AT}x_0 + \int_{\tau=0}^{T} e^{A(T-\tau)}Bu(\tau)d\tau$$

Goal: find a linear control law $u = -Kx$ such that the closed loop system

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

is stable at $x = 0$ (assumes x are coordinates relative to location of equilibrium)



- Stability based on eigenvalues ⇒ use K to make eigenvalues of $(A - BK)$ stable
- Can also link eigenvalues to *performance* (eg, initial condition response)
- Question: when can we place the eigenvalues anyplace that we want?

Theorem The eigenvalues of $(A - BK)$ can be set to arbitrary values if and only if the pair (A, B) is reachable.