control and dynamics

Prof. Sawyer B. Fuller
ME 586: Biology-inspired robotics
Project-based portion of this course

- you will work with the crazyflie helicopter as part of your homework problem sets
  - learning objectives:
    - learn basics of robotics and drone control
    - experience implementing bio-inspired control algorithm
- crazyflie specs:
  - ~30 g, ~4 minute flight time
  - communicates in real-time over bluetooth to laptop
  - sensor suite gives information needed to stabilize and control flight
  - open-source control software
three ideas inspired by biology for how to improve robotics

(the themes of this course)

1. adaptation through evolution and learning

2. mechanical intelligence
   • the use of mechanics to reduce or eliminate the need for feedback control

3. parsimony
   • simple and efficient solutions

fundamental engineering processes used by biology

“curse of dimensionality”

“shortcut”: look directly to biology for inspiration, combine with engineering knowledge

ME586 homework and projects emphasize these. We will show that the optimal control formulation we use for flight stability is also the basis for robot learning.
crazyflie in operation performing odor source localization

Odor Localization

the controller we will learn

problem set 4: using stable hover as starting point, build high-level behaviors

in this course we will learn ways to design these

model-based or model-free

model-based control for basic stability
basics: actuation for hovering

- honeybee
- single-rotor helicopter
- robot flies e.g. UW Robofly
- four-rotor aircraft “quad-rotors”

- roll torque
- pitch torque
- yaw torque
- thrust force

typically, lateral thrust is not directly actuated
If you can tilt, how do you move laterally?

\[ \dot{v}_x = f_x = f \sin \theta \approx m g \sin \theta \]
\[ \Rightarrow \dot{v}_x = g \sin \theta \approx g \theta \text{ for small } \theta \]

- "helicopter-like" lateral control
quad-rotor actuation

• two rotors spin one direction and two in the other direction

actuation with two wings

• vary angle and amplitude of flapping wings
insight into flight control: One approach is nested loops (problem set 2)

outer loop regulates position. assumes inner loop response is essentially instantaneous

inner loop regulates attitude

derivative terms add damping

• plus a separate, independent altitude controller:
more systematic and modern approach

1. A good model: Newton-Euler equations of Motion

\[ \Sigma f = m \dot{v} \]
\[ \Sigma \tau = J \dot{\omega} + \omega \times J \omega \]

• force and torque \( f, \tau \)
• linear, angular velocity \( v, \omega \)
• moment of inertia matrix \( J \)

• this is a nonlinear system.
• we will control it with linear feedback controller
• will return to these equations in more detail next week

2. Optimal control: measure performance with a cost function
Controlling nonlinear systems using linear state-space control
Model: rigid body physics

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

State-space model example: a Spring Mass System

Converting models to state space form

- Construct a vector of the variables that are required to specify the evolution of the system
- Write dynamics as a system of first order differential equations:

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

"State space form"
Simulating a state-space system

Python simulation

```python
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))

def f(q, u):
    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)
    q_data[idx, :] = q

plt.plot(time, q_data[:, 0:2])
plt.legend(('car displacement (q1)',
            'tire displacement (q2)'))
```

**basic task: repeatedly calculate state update:**

\[
q_{t+\Delta T} = q_t + \Delta T \dot{q}_t = q_t + \Delta T f(q)
\]
general form of differential equations

State space form

\[
\begin{align*}
\frac{dx}{dt} &= f(x, u) \\
y &= h(x, u)
\end{align*}
\]

General form

\[
\begin{align*}
\frac{dx}{dt} &= Ax + Bu \\
y &= Cx + Du
\end{align*}
\]

\[x \in \mathbb{R}^n, u \in \mathbb{R}^p, y \in \mathbb{R}^q\]

• \(x\) = 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:

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

$x_2 = dx_1/dt$
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

**Asymptotically stable** if all nearby initial conditions converge to the equilibrium point
- Stable + converging

**Unstable** if some initial conditions diverge from the equilibrium point
- May still be some initial conditions that converge
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”
Linearization about an equilibrium point

\[
\begin{align*}
\dot{x} &= f(x, u) \\
y &= h(x, u)
\end{align*}
\]

\[
\begin{align*}
\dot{z} &= Az + Bv \\
w &= Cz + Dv
\end{align*}
\]

to “linearize” around \(x = x_e\):

1. find \(x_e, u_e\) such that \(f = 0\)

2. define \(y_e = h(x_e, u_e)\)

\[
\begin{align*}
z &= x - x_e \\
v &= u - u_e \\
w &= y - y_e
\end{align*}
\]

3. then

\[
\begin{align*}
A &= \left. \frac{\partial f}{\partial x} \right|_{(x_e, u_e)} \\
B &= \left. \frac{\partial f}{\partial u} \right|_{(x_e, u_e)} \\
C &= \left. \frac{\partial h}{\partial x} \right|_{(x_e, u_e)} \\
D &= \left. \frac{\partial h}{\partial u} \right|_{(x_e, u_e)}
\end{align*}
\]

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
\(\implies\) nonlinear system (incl control) is stable nearby
Jacobian linearization matrix

\[ A = \left. \frac{\partial f}{\partial x} \right|_{(x_e, u_e)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \bigg|_{(x_e, u_e)} \]
Example: Stability Analysis of Inverted Pendulum

System dynamics
\[
\frac{dx}{dt} = \begin{bmatrix} x_2 \\ \sin x_1 - \gamma x_2 \end{bmatrix},
\]

Equilibria: where \( \dot{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies x_e = \begin{bmatrix} \pm \pi k, \ k = 0, 1, 2, 3... \\ 0 \end{bmatrix} \)

Linearize to assess stability:
\[
\frac{df}{dx} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \cos x_1 & -\gamma \end{bmatrix}
\]

Upward equilibria: \( x_1 = \pm 2\pi k, \ k = 0, 1, 2, 3... \)

\[
A = \frac{\partial f}{\partial x} \bigg|_{x_e} = \begin{bmatrix} 0 & 1 \\ 1 & -\gamma \end{bmatrix}
\]

eigenvalues: \( \lambda = -\frac{1}{2}\gamma \pm \frac{1}{2}\sqrt{4 + \gamma^2} \)

for \( \gamma = 0.1, \ \lambda \approx (0.95, -1.05) \implies \text{unstable} \)

Downward equilibria: \( x_1 = \pi \pm 2\pi k, \ k = 0, 1, 2, 3... \)

use \( z_1 = x_1 - x_{1e} = x_1 - \pi, \ z_2 = x_2 \implies \dot{z} = Az \)

\[
A = \frac{\partial f}{\partial x} \bigg|_{x_e} = \begin{bmatrix} 0 & 1 \\ -1 & -\gamma \end{bmatrix}
\]

eigenvalues: \( \lambda = -\frac{1}{2}\gamma \pm \frac{1}{2}\sqrt{-4 + \gamma^2} \)

for \( \gamma = 0.1, \ \lambda \approx (-0.05 + i, -0.05 - i) \implies \text{stable} \)
Model: rigid body physics

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

\[
\begin{align*}
    m_1 \ddot{q}_1 &= k_2(q_2 - q_1) - k_1 q_1 \\
    m_2 \ddot{q}_2 &= k_3(u - q_2) - k_2(q_2 - q_1) - c \dot{q}_2
\end{align*}
\]

Matrix representation:

\[
\dot{x} = Ax + Bu
\]

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

\[
y = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}
\]

“State space form”

\[
y = [1 \quad 1 \quad 0 \quad 0]x = Cx
\]
State Space Control Design Concepts

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

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

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 \( x_0, x_f \in \mathbb{R}^n \), find an input \( u(t) \) such that
  \[ \dot{x} = f(x, u(t)) \text{ takes } x(t_0) = x_0 \rightarrow x(T) = x_f \]
Tests for Reachability

\[ \begin{align*}
\dot{x} &= Ax + Bu \\
y &= Cx
\end{align*} \quad x \in \mathbb{R}^n, \quad x(0) \text{ given} \quad \begin{align*}
x(T) &= e^{AT} x_0 + \int_{\tau=0}^{T} e^{A(T-\tau)} Bu(\tau) d\tau
\end{align*} \]

**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.

**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

\[
\begin{align*}
\dot{x} &= Ax + Bu & x \in \mathbb{R}^n, \ x(0) \text{ given} \\
y &= Cx & u \in \mathbb{R}, \ y \in \mathbb{R}
\end{align*}
\]

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 \( \Rightarrow \) 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.

Next: one way to choose \( K \)
control and dynamics

Prof. Sawyer B. Fuller
ME 586: Biology-inspired robotics
Project-based portion of this course

- you will work with the crazyflie helicopter as part of your homework problem sets
  - learning objectives:
    - learn basics of robotics and drone control
    - experience implementing bio-inspired control algorithm
- Crazyflie specs:
  - ~30 g, ~4 minute flight time
  - communicates in real-time over bluetooth to laptop
  - sensor suite gives information needed to stabilize and control flight
  - open-source control software
three ideas inspired by biology for how to improve robotics

(the themes of this course)

1. adaptation through evolution and learning

2. mechanical intelligence
   - the use of mechanics to reduce or eliminate the need for feedback control

3. parsimony
   - simple and efficient solutions

😀 fundamental engineering processes used by biology
😊 “shortcut”: look directly to biology for inspiration, combine with engineering knowledge
😢 “curse of dimensionality”

ME586 homework and projects emphasize these. We will show that the optimal control formulation we use for flight stability is also the basis for robot learning.
crazyflie in operation performing odor source localization

Odor Localization

Problem set 4: Using stable hover as starting point, build high-level behaviors.

In this course, we will learn ways to design these behaviors.

First step in control design: Assume full state is known.

System dynamics:
- \( q_d \) (desired state)
- \( e_q \) (error)
- \( q \) (system state)
  - \([x, y, z \text{ position}, x, y, z \text{ velocity}, \text{roll, pitch, yaw angles}, \text{roll, pitch, yaw rates}]\)
- \( u \) (system input)
  - \([\text{thrust force}, \text{roll torque}, \text{pitch torque}]\)
- \( y \) (system output)
  - \("\text{measurement}\"")
  - \([\text{roll, pitch, yaw rates (gyro)}, \text{x-optic flow, y-optic flow (optic flow camera)}, \text{z-distance measurement (time of flight)}]\)

Estimator:
- \( \hat{q} \) (estimated state)

Controller:
- Model-based control for basic stability

High-level controller:
- or model-free

Model-based control for basic stability

University of Washington
basics: actuation for hovering

- honeybee
- single-rotor helicopter
- robot flies e.g. UW Robofly
- four-rotor aircraft “quad-rotors”

- roll torque
- pitch torque
- yaw torque
- thrust force

typically, lateral thrust is not directly actuated
If you can tilt, how do you move laterally?

lateral acceleration

\[ m\dot{v}_x = f_x = f \sin \theta \approx mg \sin \theta \]

\[ \Rightarrow \dot{v}_x = g \sin \theta \]

\[ \approx g\theta \quad \text{for small } \theta \]

- “helicopter-like” lateral control
quad-rotor actuation

actuation with two wings

- two rotors spin one direction and two in the other direction
- vary angle and amplitude of flapping wings
insight into flight control: One approach is nested loops (problem set 2)

outer loop regulates position. assumes inner loop response is essentially instantaneous

inner loop regulates attitude

• plus a separate, independent altitude controller:
more systematic and modern approach

1. A good model: Newton-Euler equations of Motion

\[ \Sigma f = m \dot{v} \]
\[ \Sigma \tau = J \dot{\omega} + \omega \times J \omega \]

- \( f, \tau \) force and torque
- \( v, \omega \) linear, angular velocity
- \( J \) moment of inertia matrix

- this is a **nonlinear** system.
- we will control it with *linear feedback controller*
- will return to these equations in more detail next week

2. Optimal control: measure performance with a cost function
Controlling nonlinear systems using linear state-space control
Model: rigid body physics

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

State-space model example: a Spring Mass System

Converting models to state space form

- Construct a vector of the variables that are required to specify the evolution of the system
- Write dynamics as a system of first order differential equations:

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

\[
y = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}
\]

"State space form"
Simulating a state-space system

```
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))
def f(q, u):
    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)
    q_data[idx, :] = q
plt.plot(time, q_data[:, 0:2])
plt.legend(('car displacement (q1)',
            'tire displacement (q2)'))
```

**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))
def f(q, u):
    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)
    q_data[idx, :] = q
plt.plot(time, q_data[:, 0:2])
plt.legend(('car displacement (q1)',
            'tire displacement (q2)'))
```
general form of differential equations

State space form

$$\frac{dx}{dt} = f(x, u)$$  $$\frac{dx}{dt} = Ax + Bu$$
$$y = h(x, u)$$  $$y = Cx + Du$$

General form  Linear system  \( x \in \mathbb{R}^n, u \in \mathbb{R}^p \)  \( y \in \mathbb{R}^q \)

\( x = \) 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:

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

**Asymptotically stable** if all nearby initial conditions converge to the equilibrium point
- Stable + converging

**Unstable** if some initial conditions diverge from the equilibrium point
- May still be some initial conditions that converge

“stable” but not asymptotically stable

asymptotically stable

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 $\lambda$ 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
  $$\dot{z} = F(x_e) + \frac{\partial F}{\partial x}\bigg|_{x_e} (x - x_e) + \text{higher order terms} \approx z = x - x_e \Rightarrow \dot{z} = Az$$

- linearization is *stable* $\Rightarrow$ nonlinear system *locally stable*
- linearization is *unstable* $\Rightarrow$ nonlinear system *locally unstable*

- “degenerate case”: if linearization is *stable* but not *asymptotically stable* $\nRightarrow$ cannot tell whether nonlinear system is stable or not!

**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

\[ \dot{x} = f(x,u) \quad \dot{z} = Az + Bu \]
\[ y = h(x,u) \quad w = Cz + Du \]

to “linearize” around \( x = x_e \):

1. Find \( x_e, u_e \) such that \( f = 0 \)

2. Define \( y_e = h(x_e, u_e) \)

3. Then

\[
A = \left. \frac{\partial f}{\partial x} \right|_{(x_e, u_e)} \quad B = \left. \frac{\partial f}{\partial u} \right|_{(x_e, u_e)} \\
C = \left. \frac{\partial h}{\partial x} \right|_{(x_e, u_e)} \quad D = \left. \frac{\partial h}{\partial u} \right|_{(x_e, u_e)}
\]

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
\( \implies \) nonlinear system (incl control) is stable nearby
Jacobian linearization matrix

\[ A = \left. \frac{\partial f}{\partial x} \right|_{(x_e, u_e)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \bigg|_{(x_e, u_e)} \]
**Example: Stability Analysis of Inverted Pendulum**

System dynamics

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

Equilibria: where \( \dot{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow x_e = \begin{bmatrix} \pm \pi k, \ k = 0, 1, 2, 3... \\ 0 \end{bmatrix} \)

Linearize to assess stability: \( \frac{df}{dx} \bigg|_{x_e} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \approx \begin{bmatrix} 0 & 1 \\ \cos x_1 & -\gamma \end{bmatrix} \)

**Upward equilibria:** \( x_1 = \pm 2\pi k, \ k = 0, 1, 2, 3... \)

\( A = \frac{df}{dx} \bigg|_{x_e} = \begin{bmatrix} 0 & 1 \\ 1 & -\gamma \end{bmatrix} \)

Eigenvalues: \( \lambda = -\frac{1}{2} \gamma \pm \frac{1}{2} \sqrt{4 + \gamma^2} \)

For \( \gamma = 0.1 \), \( \lambda \approx (0.95, -1.05) \Rightarrow \text{unstable} \)

**Downward equilibria:** \( x_1 = \pi \pm 2\pi k, \ k = 0, 1, 2, 3... \)

Use \( z_1 = x_1 - x_{1e} = x_1 - \pi, \ z_2 = x_2 \Rightarrow \dot{z} = Az \)

\( A = \frac{df}{dx} \bigg|_{x_e} = \begin{bmatrix} 0 & 1 \\ -1 & -\gamma \end{bmatrix} \)

Eigenvalues: \( \lambda = -\frac{1}{2} \gamma \pm \frac{1}{2} \sqrt{-4 + \gamma^2} \)

For \( \gamma = 0.1 \), \( \lambda \approx (-0.05 + i, -0.05 - i) \Rightarrow \text{stable} \)
Model: rigid body physics

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

\[
\begin{align*}
  m_1 \ddot{q}_1 &= k_2 (q_2 - q_1) - k_1 q_1 \\
  m_2 \ddot{q}_2 &= k_3 (u - q_2) - k_2 (q_2 - q_1) - c \dot{q}_2
\end{align*}
\]

Matrix representation:

\[
\dot{x} = Ax + Bu
\]

\[
\begin{bmatrix}
  \dot{q}_1 \\
  \dot{q}_2
\end{bmatrix} = \begin{bmatrix}
  0 & 0 \\
  0 & -k_1/m
\end{bmatrix} \begin{bmatrix}
  m \\
  k_2/m
\end{bmatrix} \begin{bmatrix}
  q_2 - q_1 \\
  q_2 - q_1
\end{bmatrix} - \begin{bmatrix}
  k_1 + k_2 \\
  k_2 + k_3
\end{bmatrix} \begin{bmatrix}
  m \\
  m
\end{bmatrix} \begin{bmatrix}
  q_1 \\
  q_2
\end{bmatrix} - c \begin{bmatrix}
  m \\
  m
\end{bmatrix} \begin{bmatrix}
  \dot{q}_1 \\
  \dot{q}_2
\end{bmatrix} + \begin{bmatrix}
  0 \\
  u
\end{bmatrix}
\]

\[
y = \begin{bmatrix}
  1 & 1 & 0 & 0
\end{bmatrix} x = Cx
\]
State Space Control Design Concepts

System description: single input, single output system (MIMO also OK)
\[ \dot{x} = f(x, u) \quad x \in \mathbb{R}^n, \ x(0) \text{ given} \]
\[ y = h(x) \quad u \in \mathbb{R}, \ y \in \mathbb{R} \]

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 \( x_0, x_f \in \mathbb{R}^n \), find an input \( u(t) \) such that
  \[ \dot{x} = f(x, u(t)) \text{ takes } x(t_0) = x_0 \rightarrow x(T) = x_f \]
Tests for Reachability

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

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

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

\[
\begin{bmatrix}
B & AB & A^2B & \ldots & A^{n-1}B \\
\end{bmatrix}
\]

is full rank.

**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

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

**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 \( \Rightarrow \) 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 anywhere 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.

Next: one way to choose \( K \)