Last Time:

- Controlled dynamics (continuous time)
- Manipulator dynamics
- Equilibria
- Stability (local)

Today:

- Continuous ODEs → Discrete-time sim
- More on stability

Motivations:

- In general, we can't solve $\dot{x} = f(x)$ for $x(t)$
- Computationally, need to represent $x(t)$ with discrete x_n
- Discrete-time models can capture some effects that continuous ODEs can't

---

Discret-Time Dynamics:

* "Explicit Form":

$$x_{n+1} = f_d(x_n, u_n)$$

↑ "discrete"

- Simplest discretization:

$$x_{n+1} = x_n + \underbrace{h f(x_n, u_n)}_{f_d(x_n, u_n)}$$

(h: time step)

} "Forward Euler Integration"

- Pendulum sim:

$$l = m = 1, \quad h = 0.1, 0.01, 0.001$$

* blows up!

Stability of Discrete-Time Systems:

- Remember, in continuous time:

$$\text{Re}[\text{eigvals}(\tfrac{\partial f}{\partial x})] < 0 \Rightarrow \text{stable}$$

- In discrete time, dynamics is an iterated map:

$$x_N = f_d(f_d(f_d(\cdots f_d(x_0))))$$

- Linearize + apply chain rule:

$$\frac{\partial x_N}{\partial x_0} = \frac{\partial f_d}{\partial x}\bigg|_{x_0} \frac{\partial f_d}{\partial x}\bigg|_{x_0} \cdots \frac{\partial f_d}{\partial x}\bigg|_{x_0} = A_d^N$$

- Assume $x = 0$ is an equilibrium

Stable $\Rightarrow \lim_{k\to\infty} A_d^k x_0 = 0 \quad \forall x_0$

$\Rightarrow \lim_{n\to\infty} A_d^n = 0$

$\Rightarrow |\text{eigvals}(A_d)| < 1$

(inside unit circle)

- Let's do this for the pendulum + forward Euler

$$x_{n+1} = \underbrace{x_n + h f(x_n)}_{f_d(x_n)}$$

$$A_d = \frac{\partial f_d}{\partial x_n} = I + hA = I + h\begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell}\cos(\theta) & 0 \end{bmatrix}$$

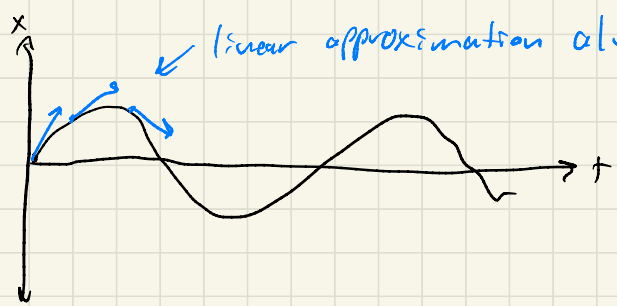
$$\text{eigvals}(A_d|_{\theta=0}) \approx 1 \pm 0.313\,i$$

$(h = 0.5)$

- Plot $|\text{eigvals}(A_d)|$ vs. h

$\Rightarrow$ Only (marginally) stable as $h \to 0$

+ Intuition:



* Take-away messages:
  - Be careful when discretizing ODEs
  - Sanity check eg energy, momentum behavior
  - Don't use forward Euler integration!

* A better explicit integrator:
  - 4TH-Order Runge-Kutta Method ("industry standard")

- Intuition
  - Euler fits a line segment over each time step
  - RK4 fits a cubic polynomial

  $\Rightarrow$ much better accuracy!

- Pseudo Code:

$$x_{n+1} = f_{rk}(x_n)$$

$$k_1 = f(x_n)$$
$$k_2 = f(x_n + h/2\, k_1)$$
$$k_3 = f(x_n + h/2\, k_2)$$
$$k_4 = f(x_n + h\, k_3)$$

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

* Take Away:

- Accuracy win >> additional compute cost
- Even "good" integrators have issues

  ⇒ always sanity check

* "Implicit Form":

$$f_d(x_{n+1}, x_n, u_n) = 0$$

- Simplest version:

$$x_{n+1} = x_n + hf(x_{n+1})$$ ← "Backward Euler"

↑ evaluate f at future time

- How do we simulate?

  write as:

$$f_d(x_{n+1}, x_n, u_n) = x_n + hf(x_{n+1}) - x_{n+1} = 0$$

- Solve root-finding problem for x_{n+1}

  move on this next week

- Pendulum Sim:
  - Opposite energy behavior from forward Euler
  - Artificial damping

- While unphysical, this allows simulators to take big steps and is often convenient

  => Very common in low-fi simulators in graphics/robotics

Take-Aways:

- Implicit methods typically "more stable" than explicit
- For forward sim, implicit methods are generally more expensive
- In many trajectory optimization methods they're not more expensive.