Last Time:

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

Today:

- Continuous ODEs → Discrete time sim
- More on stability

Motivation:

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

---

Discrete-Time Dynamics:

* "Explicit" Form:

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

↑ "discrete"

- Simplest discretization:

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

h: time step

f_d

≈ "Forward Euler Integration"

- Pendulum Simulation

$l = m = 1$, $h = 0.1$, 0.01

(blows up)

Stability of Discrete-Time Systems:

- Remember, in continuous time:

$$\text{Re}\left[\text{eig}\left(\frac{\partial f}{\partial x}\right)\right] < 0 \Rightarrow \text{stable}$$

- 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:

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

- Assume we set up coordinates so equilibrium is at $x_0 = 0$

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

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

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

(inside unit circle)

- Let's apply this to pendulum + forward Euler

$x_{k+1} = \underbrace{x_k + hf(x_k)}_{f_d(x_k)}$

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

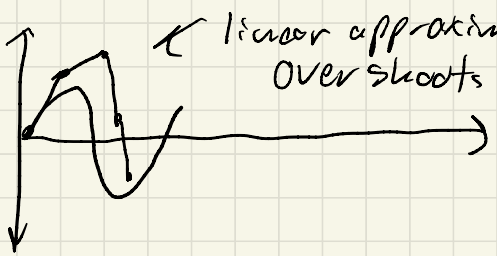
$\text{eig}(A_d|_{\theta=0}) \approx 1 \pm 0.313i$

$(h = 0.1)$

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

$\Rightarrow$ Only (marginally) stable in limit $h \to 0$

- Intuition:



* Take Away Messages:
  - Be careful when discretizing ODEs
  - Sanity check based on e.g. energy, momentum behavior (conservation and/or dissipation)
  - 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
      ⇒ much better accuracy!
  - Pseudo Code:

$$x_{n+1} = f_{RK4}(x_n)$$

$$k_1 = f(x_n)$$
$$k_2 = f(x_n + \tfrac{h}{2} k_1)$$
$$k_3 = f(x_n + \tfrac{h}{2} k_2)$$
$$k_4 = f(x_n + h k_3)$$

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

- Take away:
  - Accuracy win >> additional compute cost
  - Even sophisticated integrators have issues

⇒ Always sanity check!

* "Implicit" form:

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

- Simplest version:

$$x_{n+1} = x_n + h f(x_{n+1})$$ = "Backward Euler"

↖ evaluate f at future time

- How do we simulate?
  - write as:

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

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

more on this next week

- Pendulum Sim:
  - Opposite energy behavior from forward Euler

- Discretization adds damping
- While unphysical, this effect allows simulators to take big steps and is often convenient

  ⇒ very common in low-fi simulators in graphics/robotics

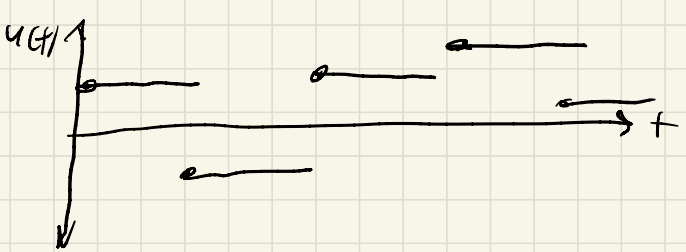
* Take Aways:
  - Implicit methods are often "more stable" than
  - For forward simulation, solving implicit equation can be more expensive
  - In many "direct" trajopt methods they're not more expensive.

Discretizing Controls:

- So far we've discretized $x(t)$
- we also have $u(t)$

* Simplest option:

$$u(t) = u_n, \quad t_n \le t < t_{n+1}$$ ← "zero-order hold"

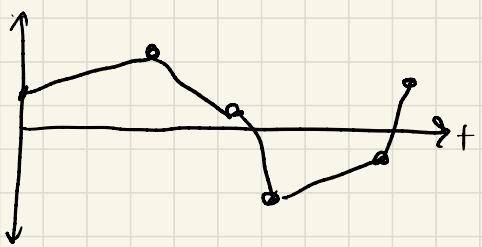


- easy to implement
- May require lots of knot points to accurately capture continuous $u(t)$

* (Possibly) Better Option

$$u(t) = u_n + \left(\frac{u_{n+1} - u_n}{h}\right)(t - t_n)$$

↖ "First-order hold"



- Can approximate continuous $u(t)$ with fewer knot points
- Not much extra work vs. zero-order hold
- Super common (e.g. classic DTRCOL)

* Other Options
  - We can keep playing this game with higher-order polynomials
  - In many control applications $u(t)$ is <u>not</u> smooth (e.g. bang-bang). Therefore high-order polynomials are not good approximations

⇒ Zero-order and first-order hold are most common in practice