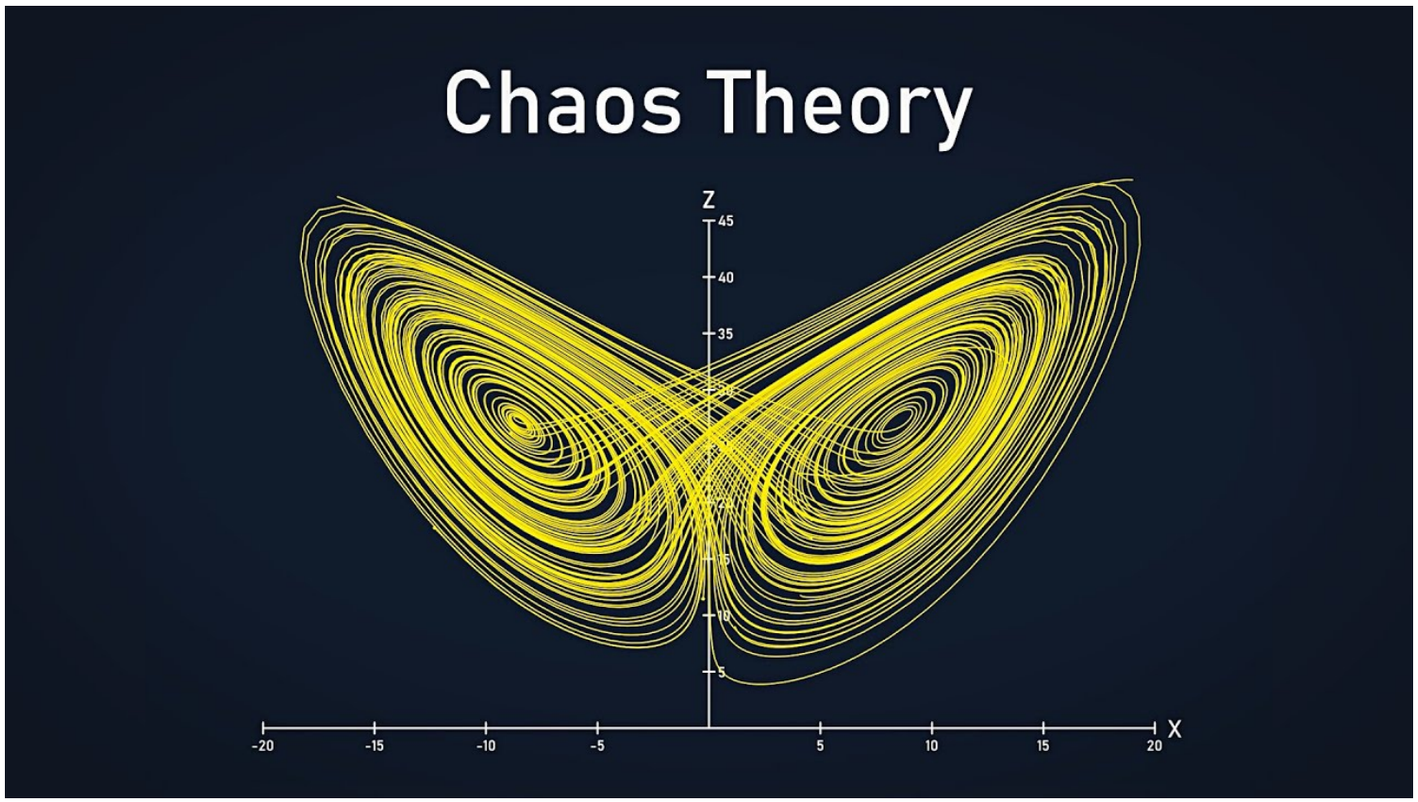
Chaos
Lecture 5



PHYS 246 class 4

Fall 2025

J Noronha-Hostler

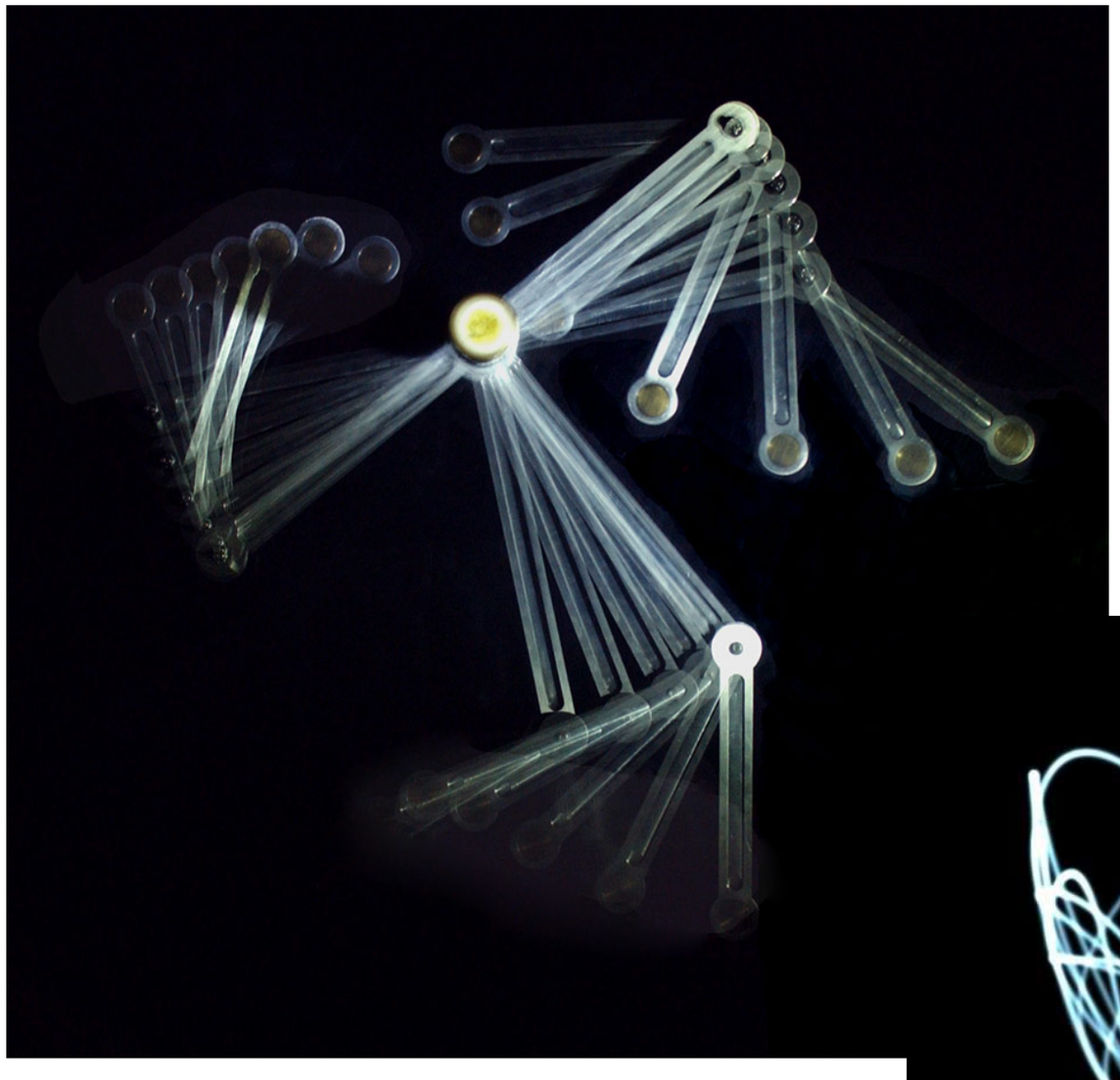
https://jnoronhahostler.github.io/IntroductionToComputationalPhysics/intro.html

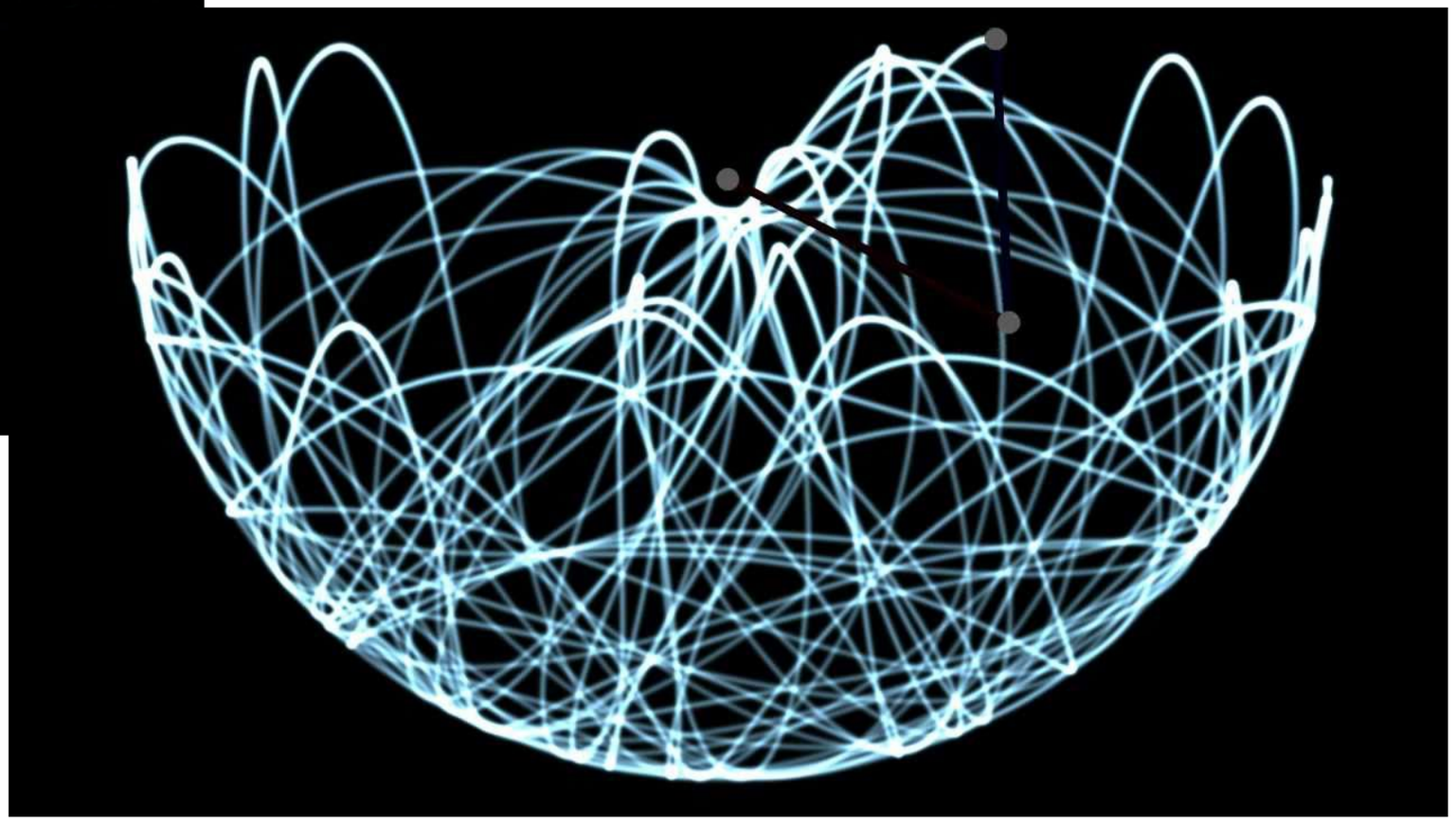
Announcements/notes

- There are still some issues with the PDFs, starting for tonight's assignment (assigned last week), you WILL lose points (10% off)!
-

Chaos and pendulums

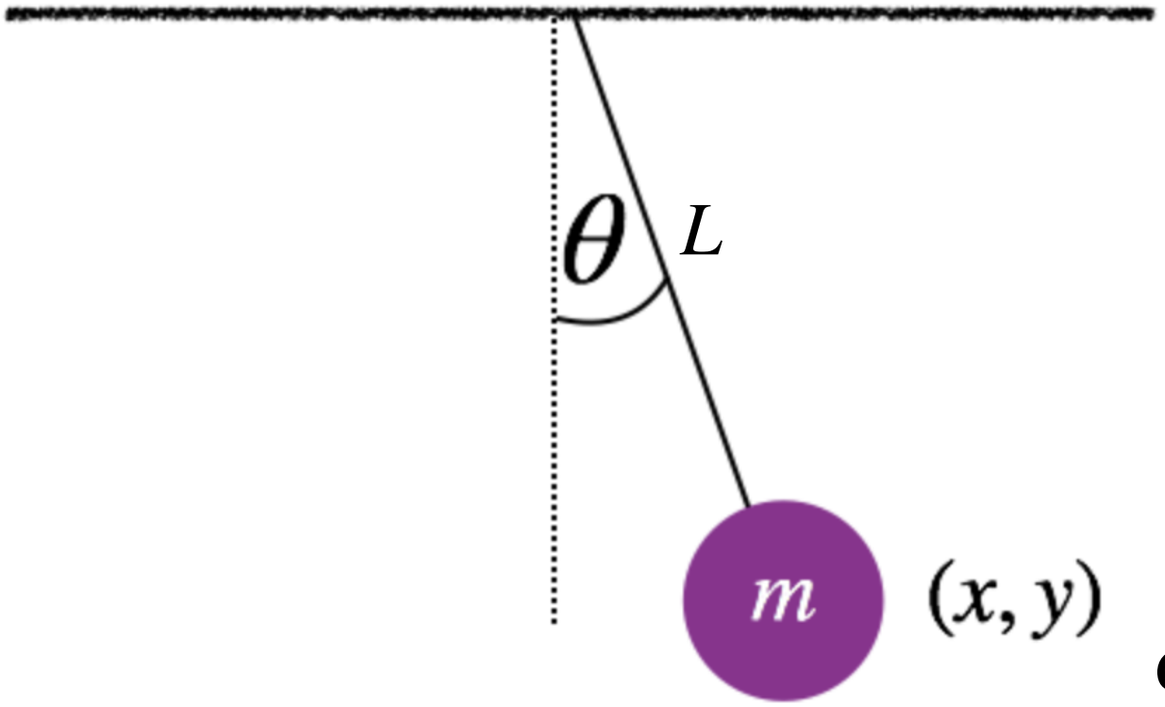
Starting from one pendulum and diving into chaos!





Pendulum

Going from $(x, y) \rightarrow \theta$



Bottom of arc

$\theta = 0$

Angular acceleration

$$\frac{d\omega}{dt} = \alpha(\theta) = -\frac{g\sin(\theta)}{L}$$

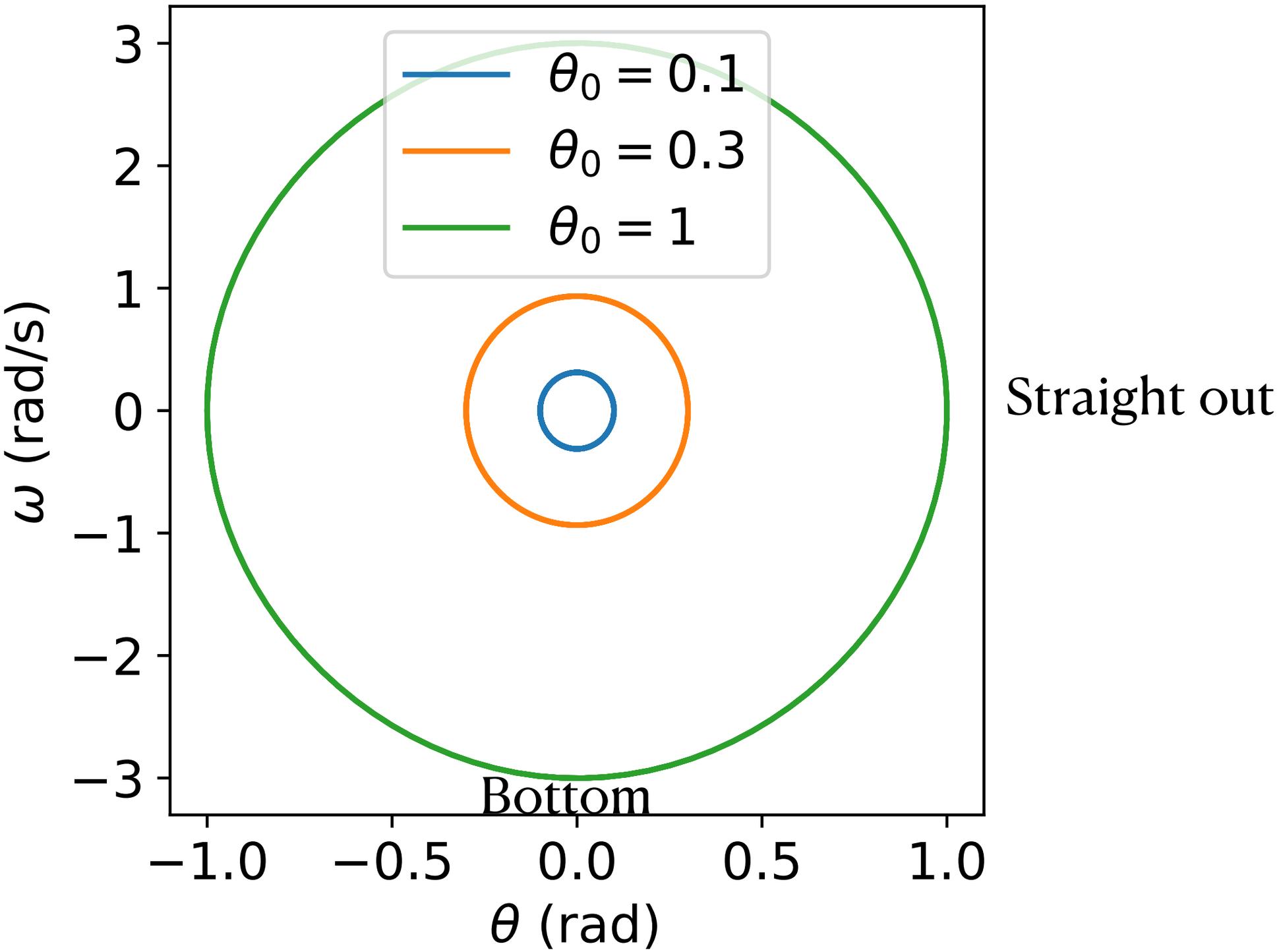
Angular velocity

$$\omega(t + \Delta t) \sim \omega(t) + \frac{d\omega}{dt}\Delta t$$

Angular displacement

$$\theta(t + \Delta t) \sim \theta(t) + \omega\Delta t$$

Angular velocity vs displacement



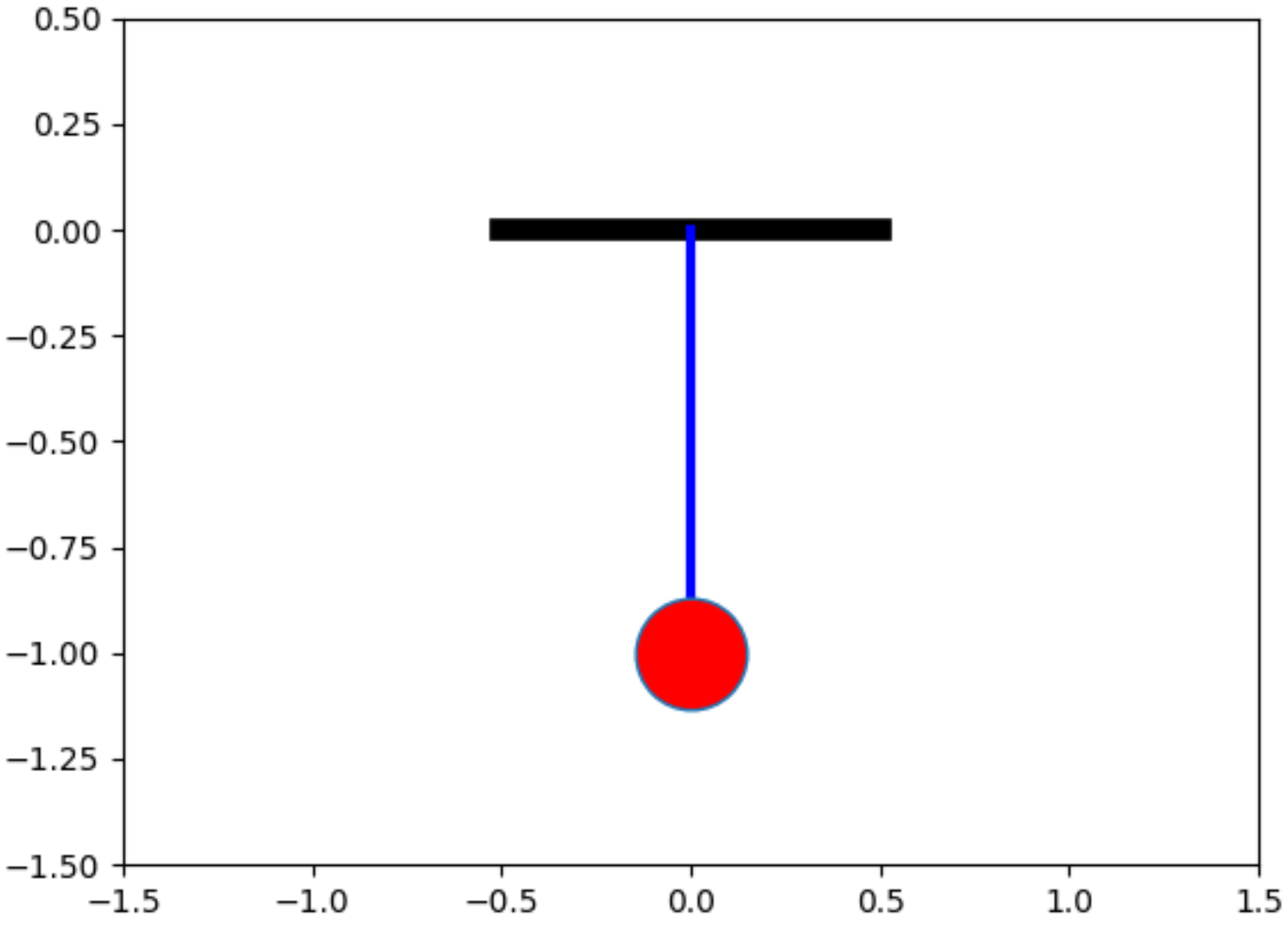
Small Angle approximation

When does it break down?

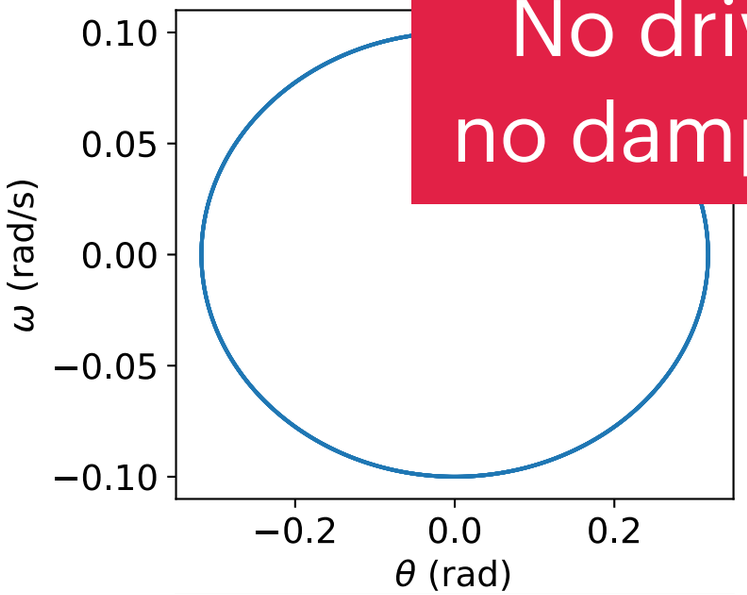
- Small angle approximation: $\theta \sim sin\theta$
  - Sanity check: $\theta = 0, \quad 0 = \sin 0$
- Can compare:
  - $$\alpha(\theta) = -\frac{g\sin(\theta)}{L} \qquad \text{vs} \qquad \alpha(\theta) = -\frac{g\theta}{L}$$
- When does this break down?

Driving Pendulum

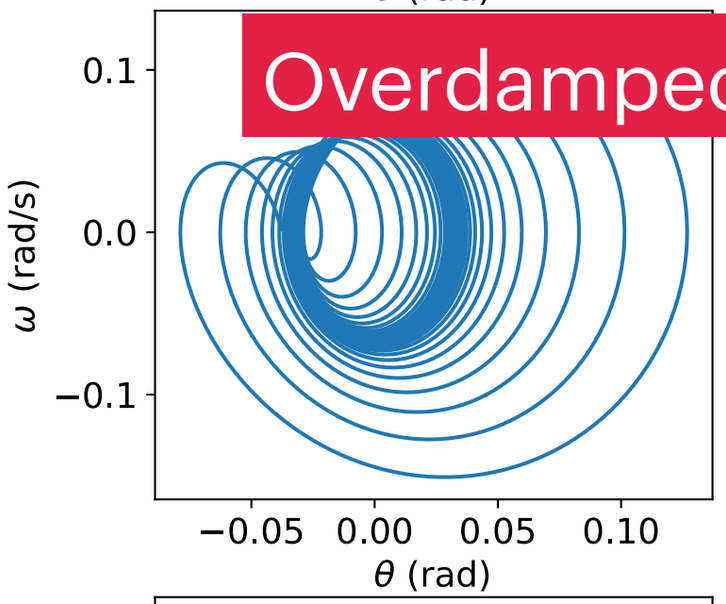
Overdamped



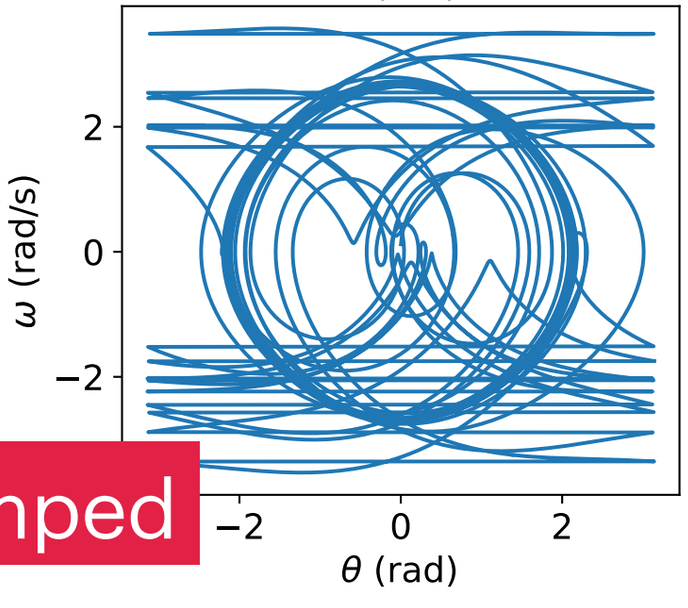
No drive, no damping



Overdamped



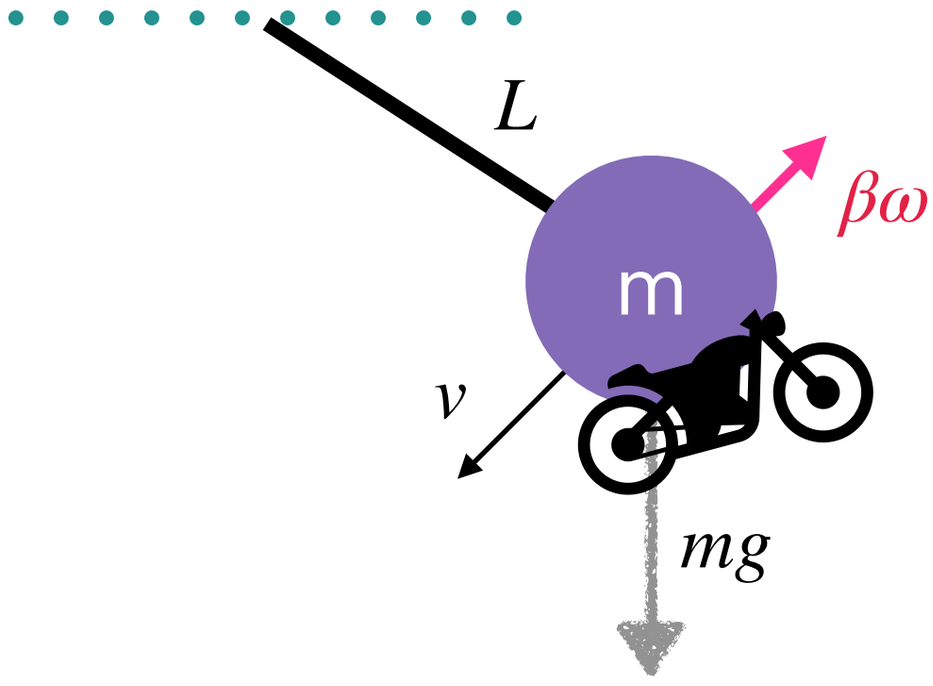
Underdamped



Underdamped (goes above start position, then slows)

Damped vs driven a pendulum

Attach ball of mass m to a rod (not string!) of length L



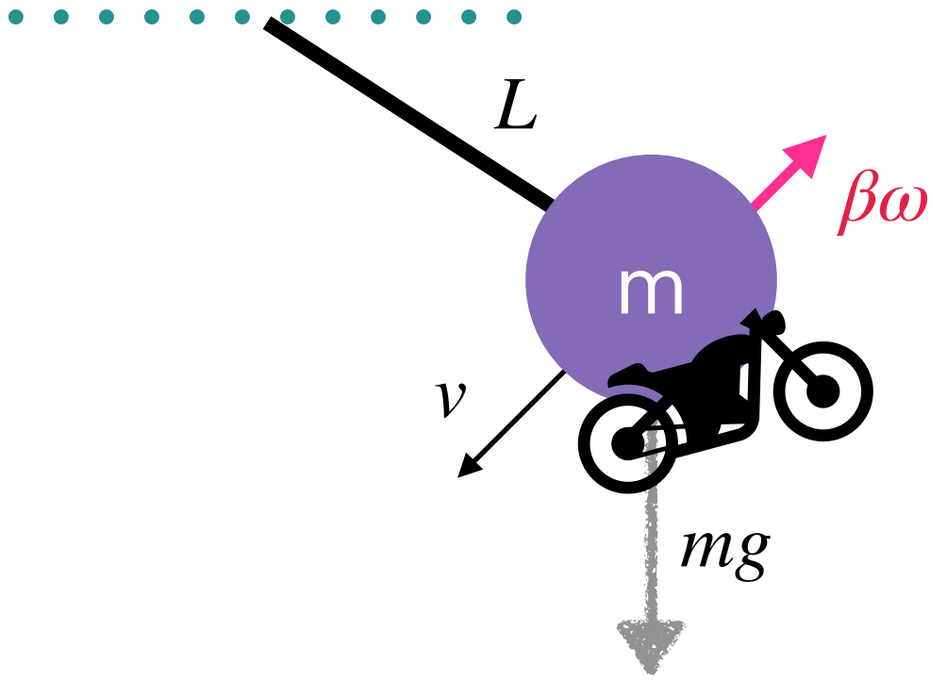
There's an angular velocity-dependent drag force $\beta\omega$ from air resistance

Pretend some engine is attached that provides an additional force (torque) $\tau = \gamma \sin \Omega_{ext} t$

$$\tau = I\alpha = mL^2 \frac{d^2\theta}{dt^2}$$

$$\frac{d\omega}{dt} = -A\omega - B\sin\theta + C\sin\Omega_{ext}t$$

Damped vs driven a pendulum



$$\tau = I\alpha = mL^2\frac{d^2\theta}{dt^2}$$

$$= \underbrace{-mgL\sin\theta}_{F_g} + \underbrace{-\beta\frac{d\theta}{dt}}_{F_{drag}} + \underbrace{\gamma\sin\Omega_{ext}t}_{F_{ext}}$$

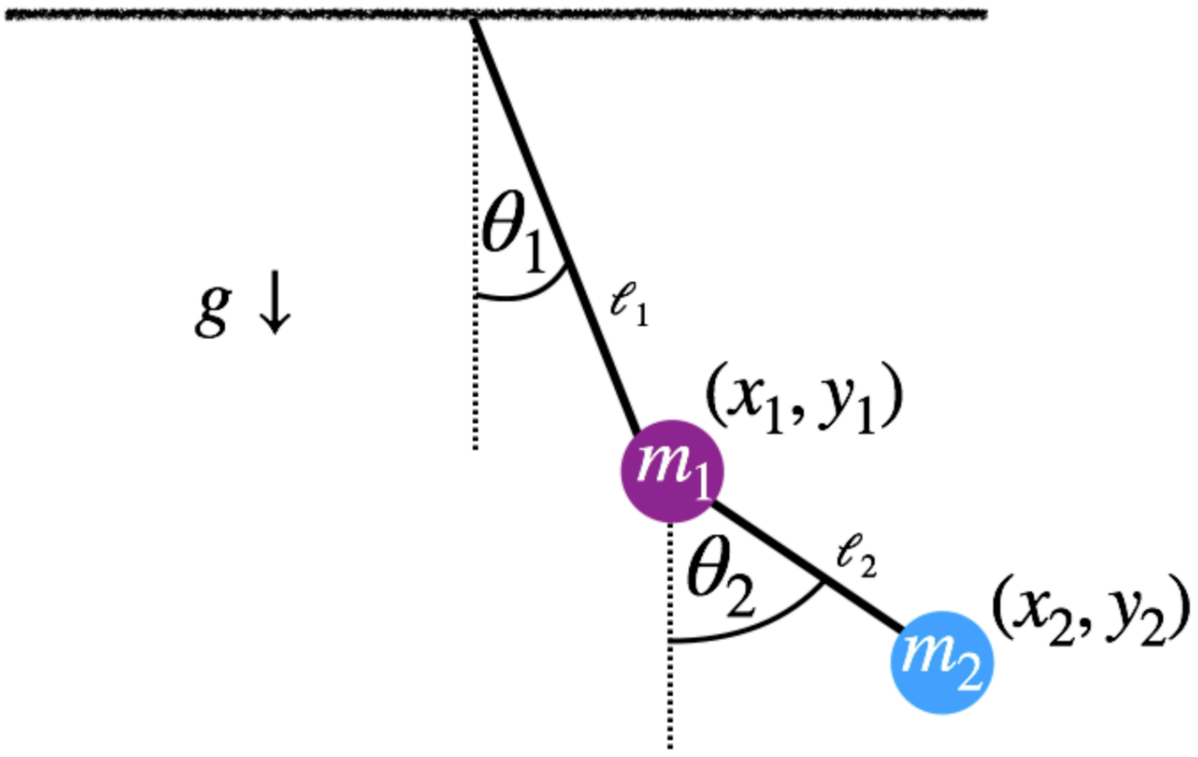
After some algebra...

$$\frac{d\omega}{dt} = -A\omega - B\sin\theta + C\sin\Omega_{ext}t$$

$$\omega = \frac{d\theta}{dt}$$

Double Pendulum

Yikes!



$$\alpha_1(\theta_1, \theta_2) := \frac{l_2}{l_1}\left(\frac{m_2}{m_1 + m_2}\right)\cos(\theta_1 - \theta_2)$$

$$\alpha_2(\theta_1, \theta_2) := \frac{l_1}{l_2}\cos(\theta_1 - \theta_2)$$

$$f_1(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) := -\frac{l_2}{l_1}\left(\frac{m_2}{m_1 + m_2}\right)\dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - \frac{g}{l_1}\sin\theta_1$$

$$f_2(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) := \frac{l_1}{l_2}\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - \frac{g}{l_2}\sin\theta_2$$

$$g_1 := \frac{f_1 - \alpha_1 f_2}{1 - \alpha_1\alpha_2}$$

$$g_2 := \frac{-\alpha_2 f_1 + f_2}{1 - \alpha_1\alpha_2}$$

A bit difficult to work out forces!

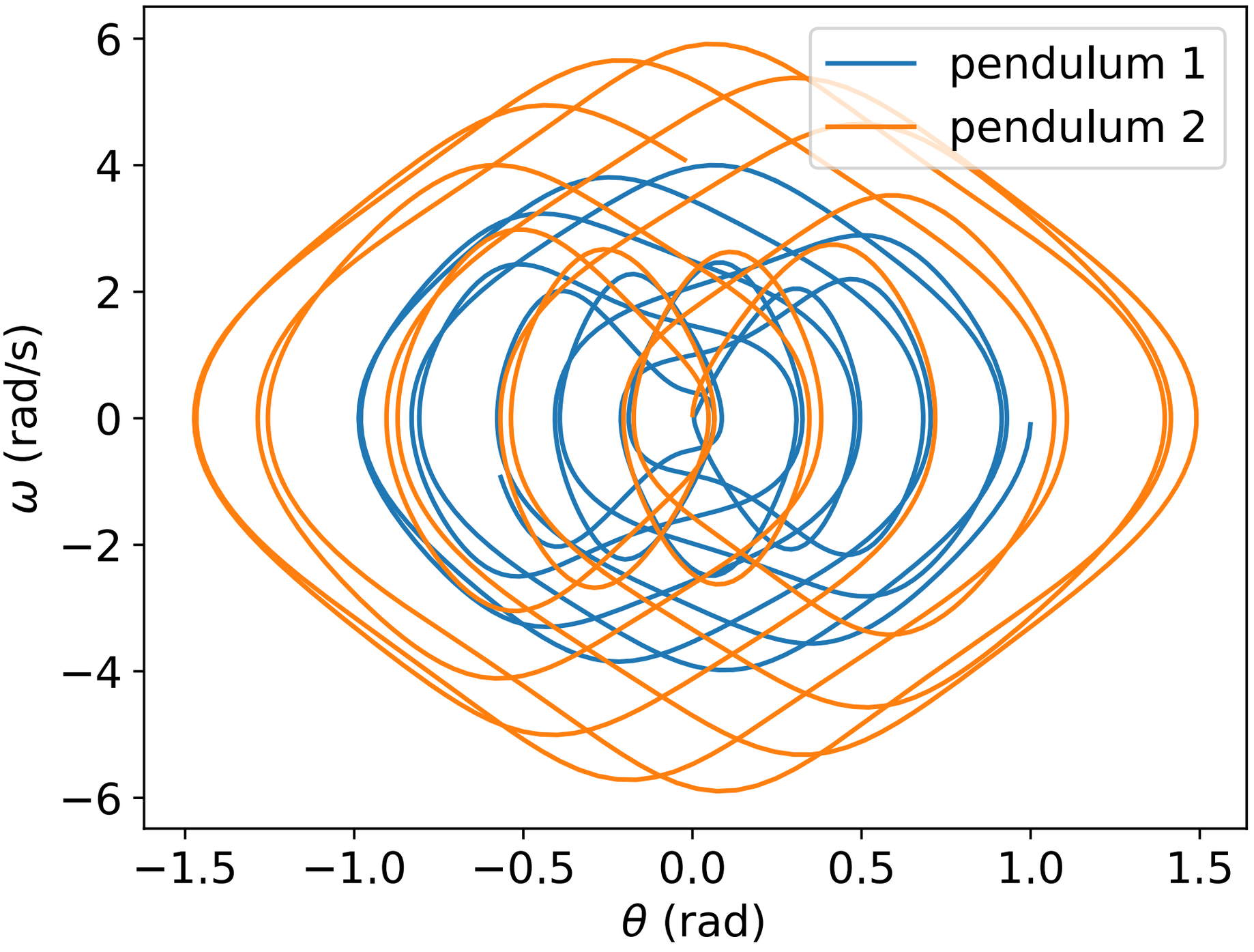
Easier to use Euler-Lagrange method (PHYS 325) (principle of least action)

$$\frac{d}{dt}\begin{pmatrix}\theta_1 \\ \theta_2 \\ \omega_1 \\ \omega_2\end{pmatrix} = \begin{pmatrix}\omega_1 \\ \omega_2 \\ g_1(\theta_1, \theta_2, \omega_1, \omega_2) \\ g_2(\theta_1, \theta_2, \omega_1, \omega_2)\end{pmatrix}$$

https://dassencio.org/33

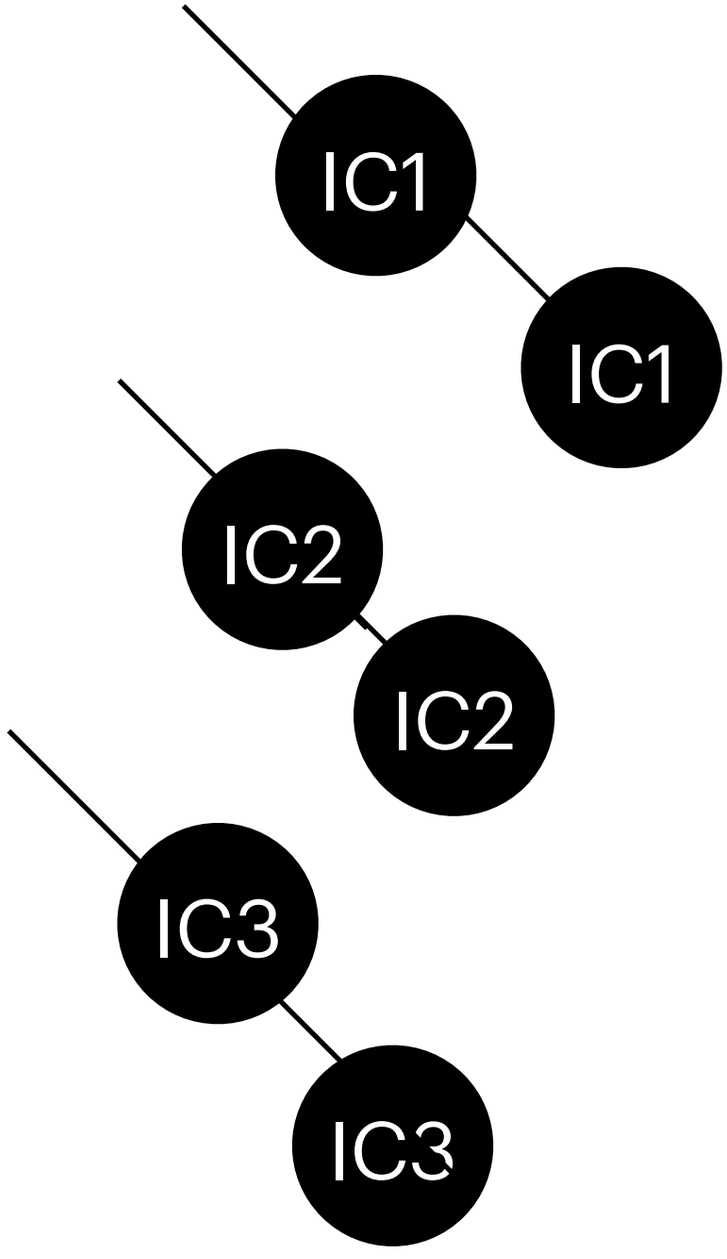
Double pendulum

Phase space

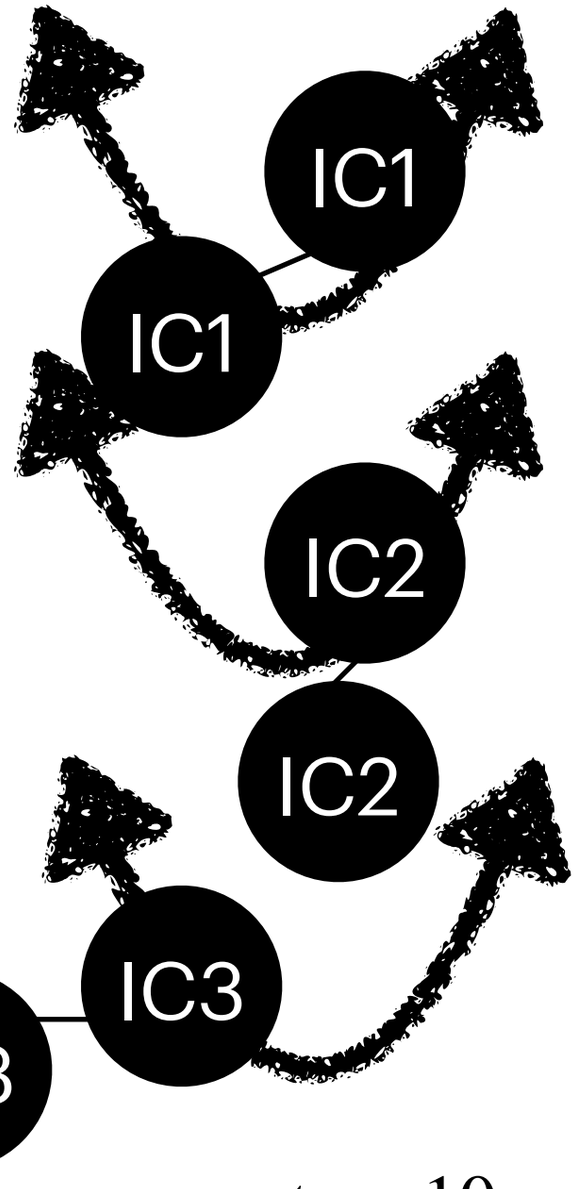


Chaos

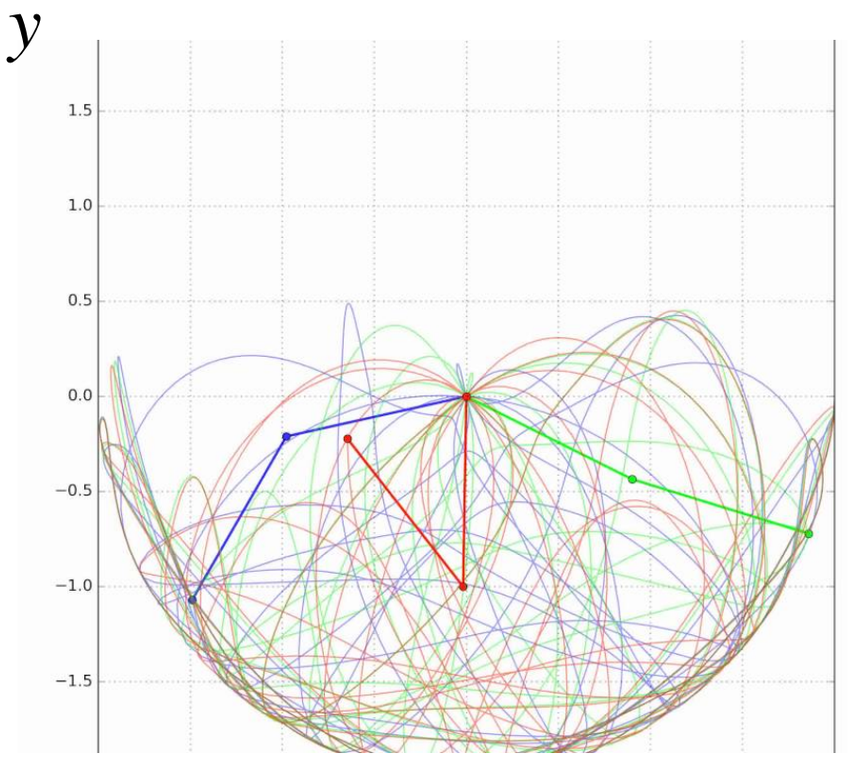
Initial conditions matter (a lot)!



t_0



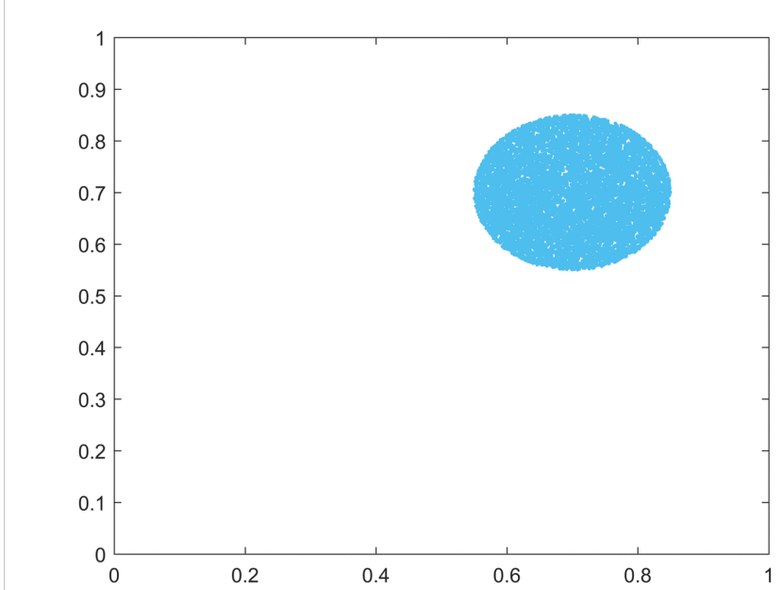
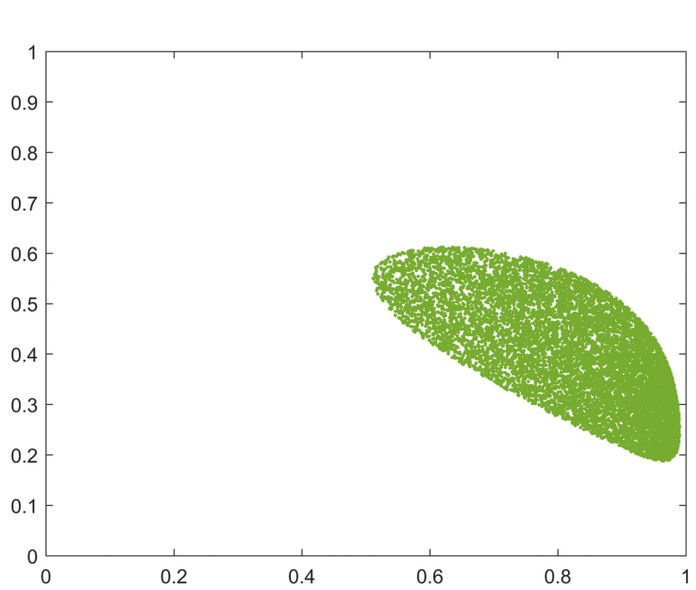
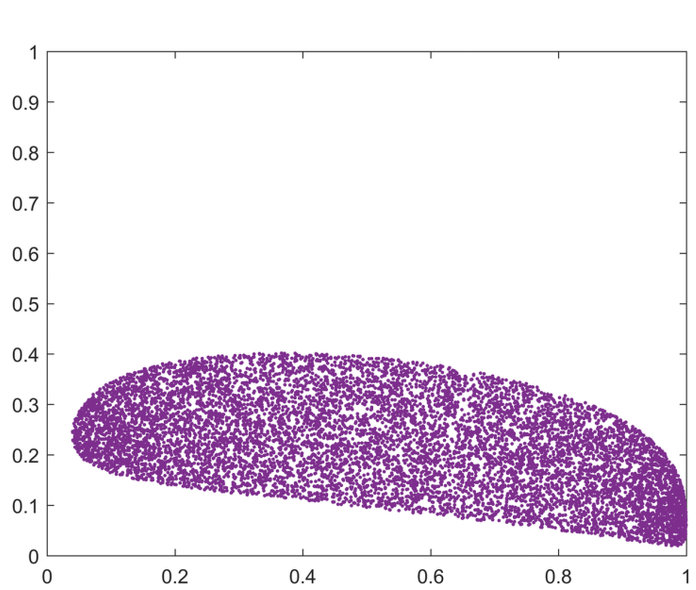
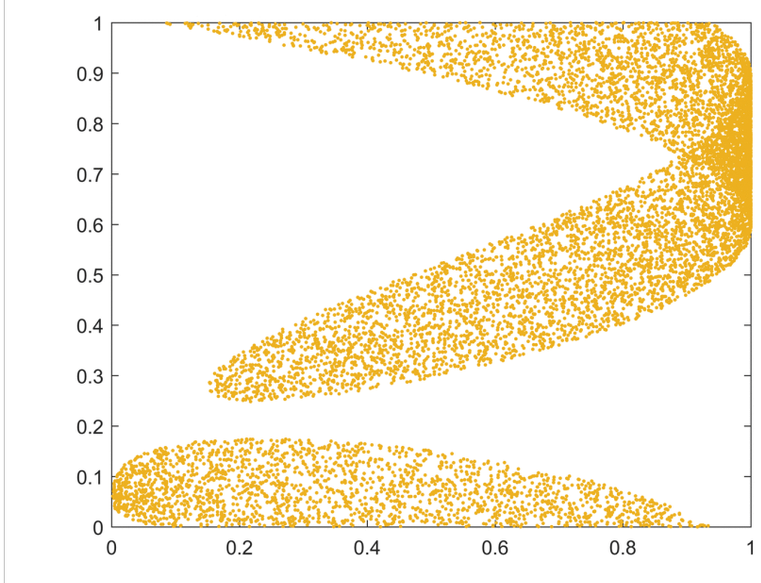
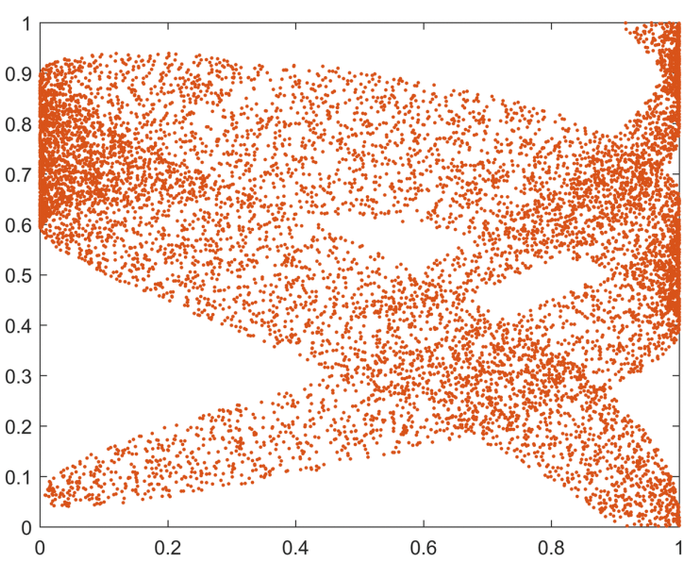
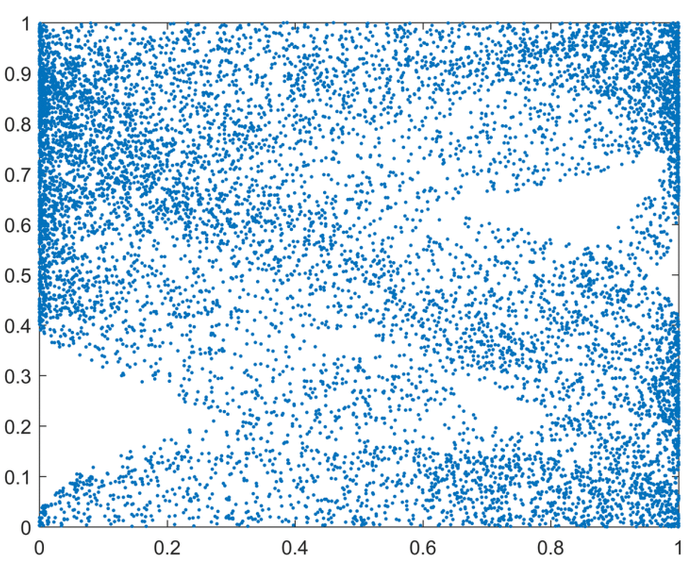
$t_0 = 10$



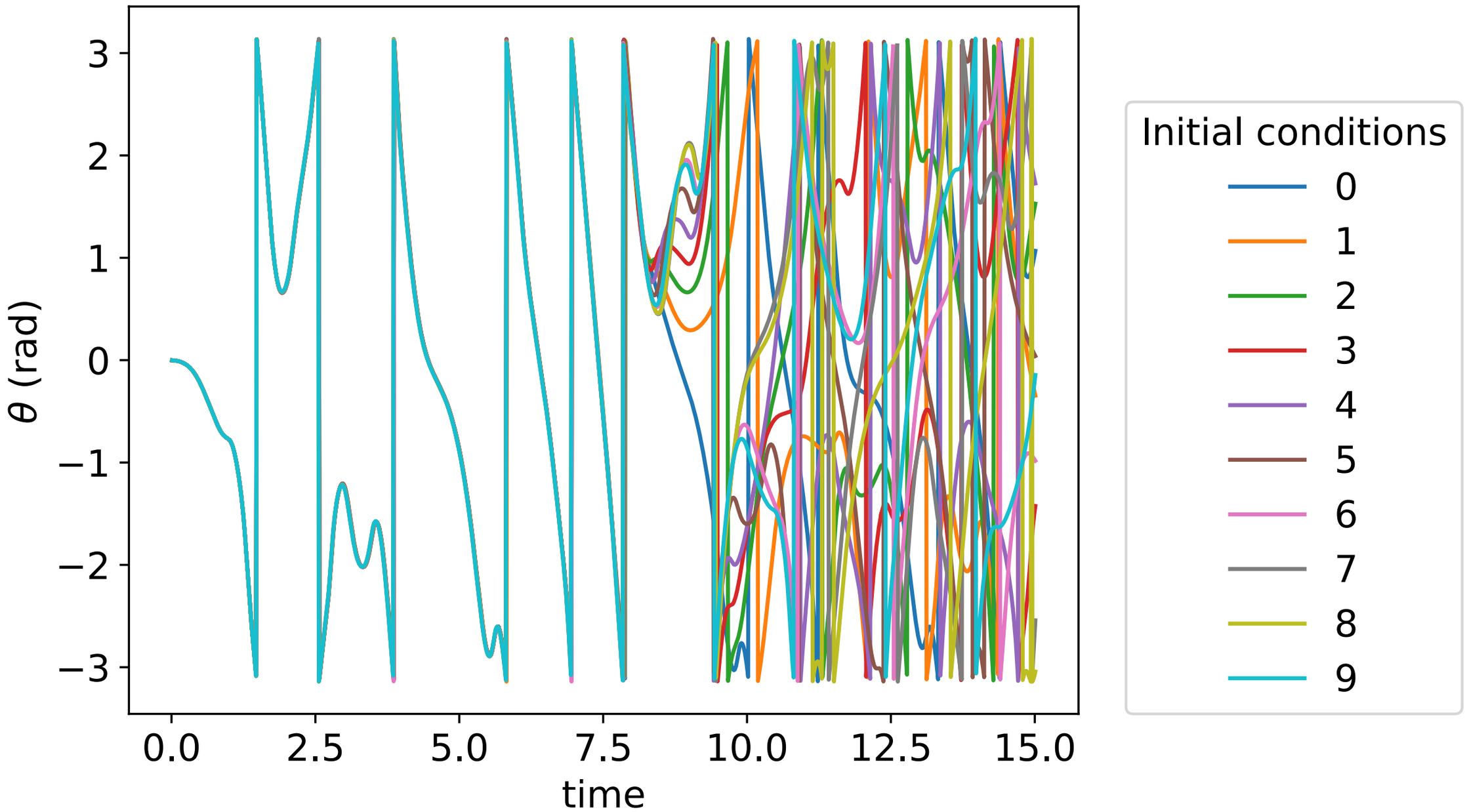
$t_0 = 100$

Chaos - general concept

By N3kromancer732 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50379199

Initial conditions over time



Checks for your code

Undamped, undriven pendulum:

- Small angles should match analytic result
- Energy should be conserved (phase space should be an ellipse)

Driven pendulum

Large driving force should make it unstable.

Small driving force plus damping --> should go to (0,0)

Double pendulum

Small initial angles/velocities, small angle approximation should still hold

Python suggestions

animate is meant to visualize multiple runs

```
def animateMe_single_pendulum(positions, params):
    """
    positions [run, time, theta]
    params {'l1': length of the pendulum}
    """
```

Magically wrap the positions to (-pi,pi)

```
    pos = pos - 2*np.pi*np.ceil((pos-np.pi)/(2*np.pi))
```

Problems with global variables

```
[7]  a=5
     ✓ 0.0s
```

```
     def f(x):
         return x * a

     print(f(2))
```

More...

```
[6]  ✓ 0.0s
...  14
```

```
[5]  a=7
     ✓ 0.0s
```

```
[8]  print(f(2))
     ✓ 0.0s
...  10
```