Student names: . . . (please update)

Instructions: Update this file (or recreate a similar one, e.g. in Word) to prepare your answers to the questions. Feel free to add text, equations and figures as needed. Hand-written notes, e.g. for the development of equations, can also be included e.g. as pictures (from your cell phone or from a scanner). ***This lab is not graded. However, the lab exercises are meant as a way to familiarise with dynamical systems and to study them using Python to prepare you for the final project.*** *This file does not need to be submitted and is provided for your own benefit. The graded exercises will have a similar format.*

The file `lab#.py` is provided to run all exercises in Python. Each `exercise#.py` can be run to run an exercise individually. The list of exercises and their dependencies are shown in Figure 1. When a file is run, message logs will be printed to indicate information such as what is currently being run and and what is left to be implemented. All warning messages are only present to guide you in the implementation, and can be deleted whenever the corresponding code has been implemented correctly.

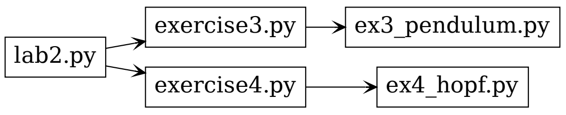


Figure 1: Exercise files dependencies. In this lab, you will be modifying `exercise3.py`, `ex3_pendulum.py`, `exercise4.py` and `ex4_hopf.py`.

Question 3: Pendulum with friction

3.a Find the fixed points of the pendulum with friction (i.e. damping), and analyze their stability using a local linearization (briefly describe the calculation steps).

$$\ddot{\theta} = -\frac{g}{L}\sin\theta - d\dot{\theta} \tag{1}$$

where θ is the angle, g the gravity constant, L the length of the pendulum and d is the damping coefficient.

Fixed points:

$$\begin{aligned} \dot{x}_2 &= -\frac{g}{L}\sin x_1 - dx_2 \qquad & sin(\tilde{x}_1) = 0 \quad &\Rightarrow \quad \tilde{x}_1 = n\pi, \quad n \in Z \\ \dot{x}_1 &= x_2 = 0 & \tilde{x}_2 = 0 & \end{aligned} \tag{2}$$

Jacobian and eigenvalues:

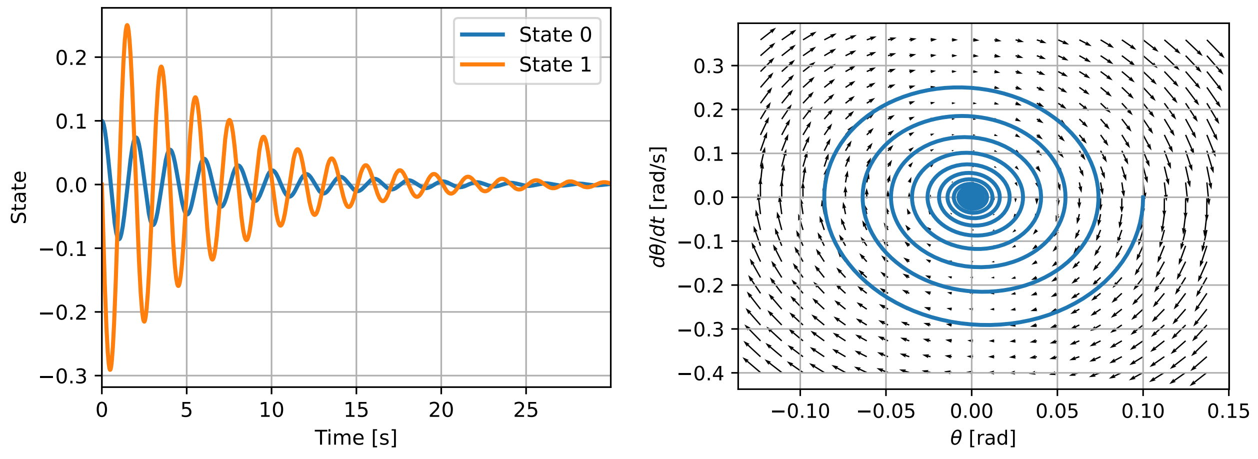
$$J = \begin{pmatrix} 0 & 1 \\ -\frac{g}{L}\cos x_1 & -d \end{pmatrix} \quad \Rightarrow \quad \lambda^2 + d\lambda + \frac{g}{L}\cos x_1 = 0 \quad \Rightarrow \quad \lambda_\pm = \frac{-d \pm \sqrt{d^2 - 4\frac{g}{L}\cos x_1}}{2} \tag{3}$$

In the case where $\tilde{x}_1 = 0$ (i.e. pendulum down), both eigenvalues $\lambda_\pm < 0$, thus the fixed point is stable. In the case $\tilde{x}_1 = \pi$, we have $\lambda_- < 0$ and $\lambda_+ > 0$, thus the fixed point is unstable (saddle point).

3.b Numerically solve the differential equations of the pendulum with different initial conditions. Show several time evolutions and phase portraits with different initial conditions that illustrate several aspects of the interesting behavior of the pendulum. See `exercise3.py` and `ex3_pendulum.py` for help with implementation.

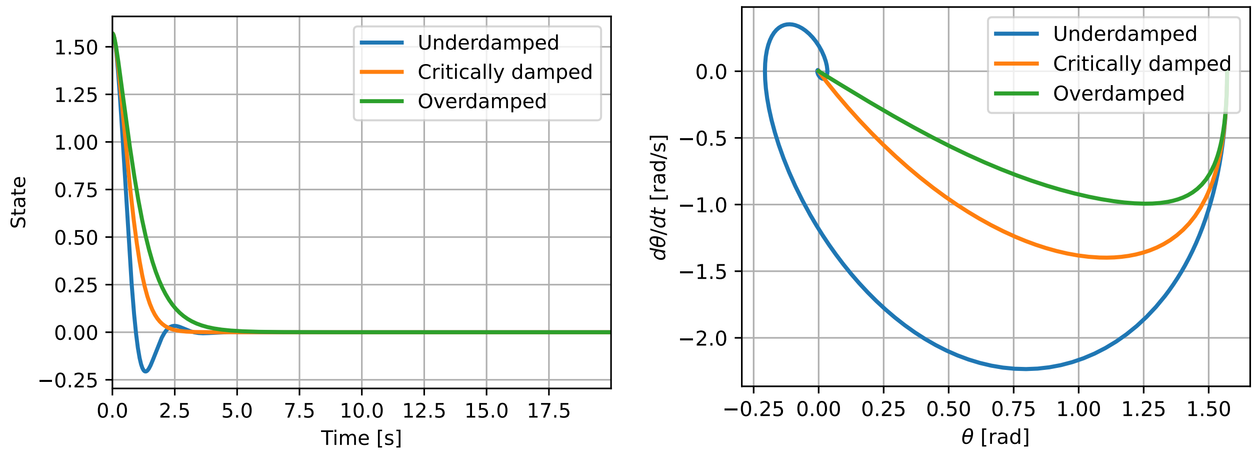
Different time evolutions should be shown. Ideally examples should include: starting and staying at unstable fixed point, simple damped oscillations around stable fixed point, complete loops due to initial high velocity that then end up at stable fixed point. Depending on the choice of the parameters, oscillations might disappear in favour of an exponential decay.

For instance:



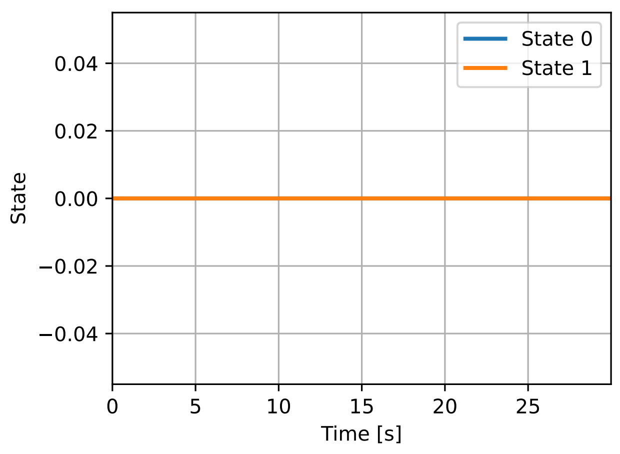
(a) Time evolution *(b) Phase portrait*

Figure 2: Basic pendulum setup ($\theta_0 = 0.1$ [rad], $\dot{\theta}_0 = 0$ [rad/s])

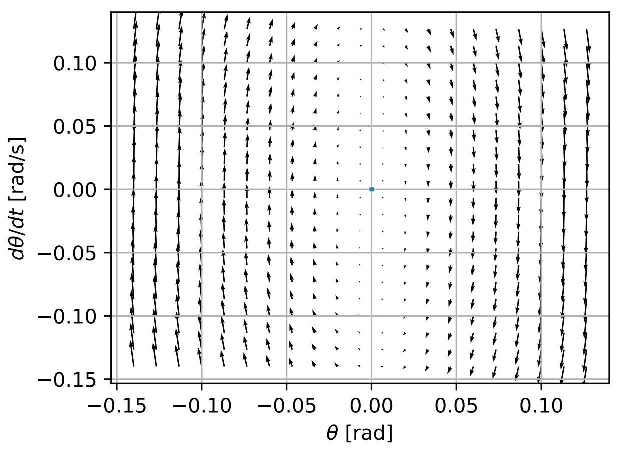


(a) Time evolution *(b) Phase portrait*

Figure 3: Different types of fixed point are obtained depending on the sign of $\Delta = d^2 - 4\frac{g}{L}$. Note that the underdamped solution shows an overshoot before settling at the equilibrium point. Also note that the critically damped solution decays faster than the overdamped (and underdamped) one.

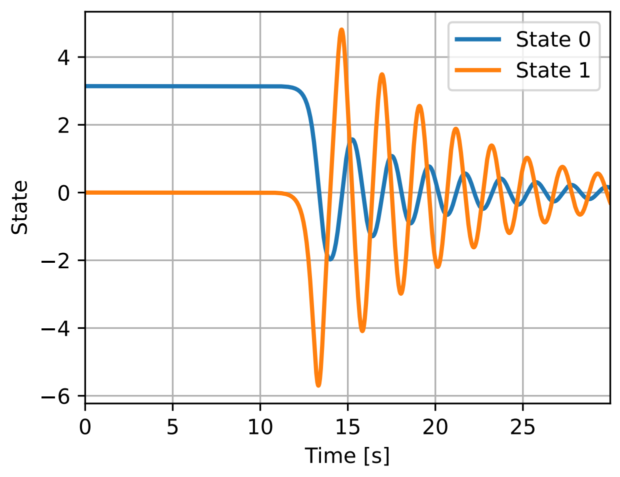


(a) Time evolution

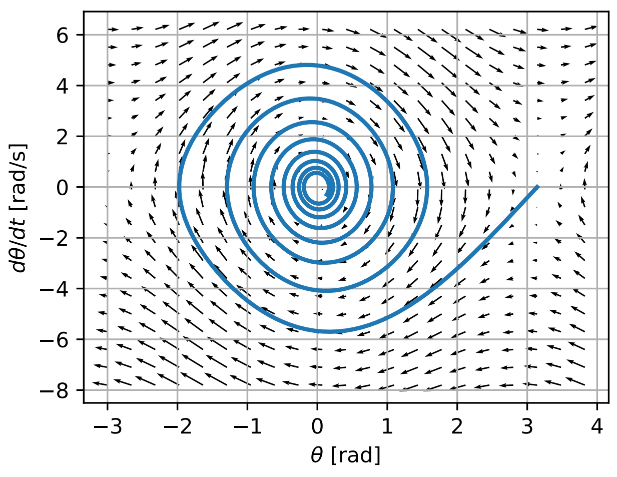


(b) Phase portrait

Figure 4: Stable pendulum setup ($\theta_0 = 0$ [rad], $\dot{\theta}_0 = 0$ [rad/s])

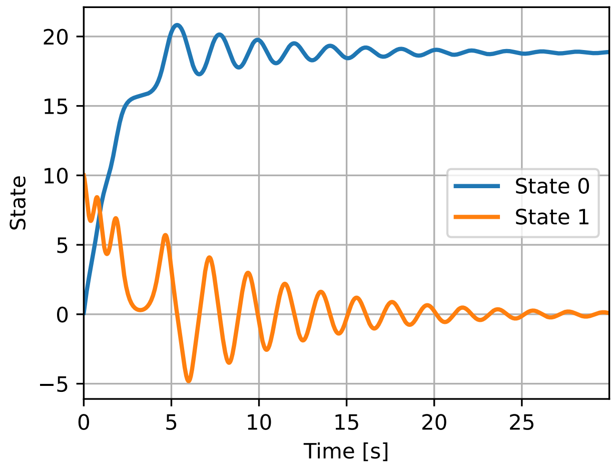


(a) Time evolution

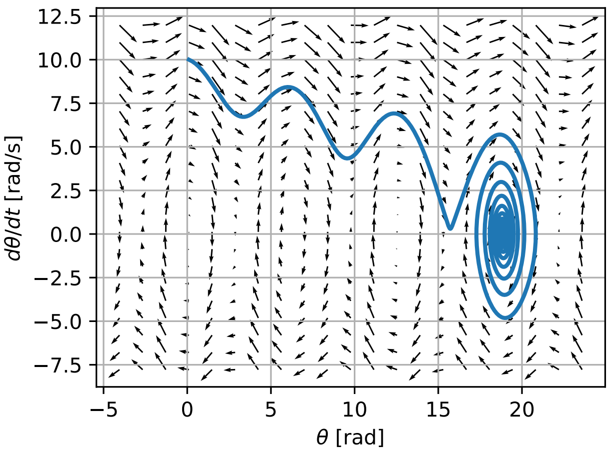


(b) Phase portrait

Figure 5: Unstable pendulum setup ($\theta_0 = \pi$ [rad], $\dot{\theta}_0 = 0$ [rad/s]). The pendulum stays at the unstable fixed point for a few seconds, and then is pushed away due to numerical imprecision of the integration. It ends up at the stable fixed point (0,0)



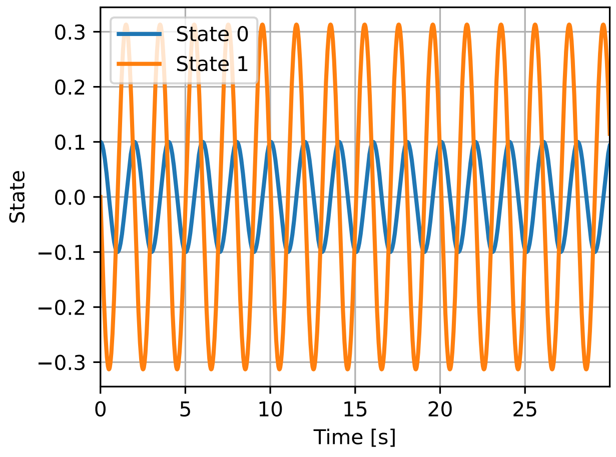
(a) Time evolution



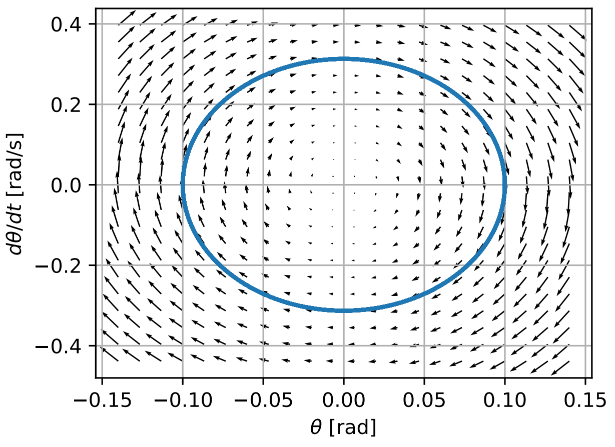
(b) Phase portrait

Figure 6: High initial velocity pendulum setup (θ_0 = 0.1 [rad], $\dot{\theta}_0$ = 10 [rad/s]). In this case, the pendulum has a high initial velocity, and therefore makes three complete rotations before converging to the stable fixed point (6 pi, 0).

3.c Investigate and describe how the behavior of the pendulum changes if friction is zero (d=0). Show a new phase portrait.



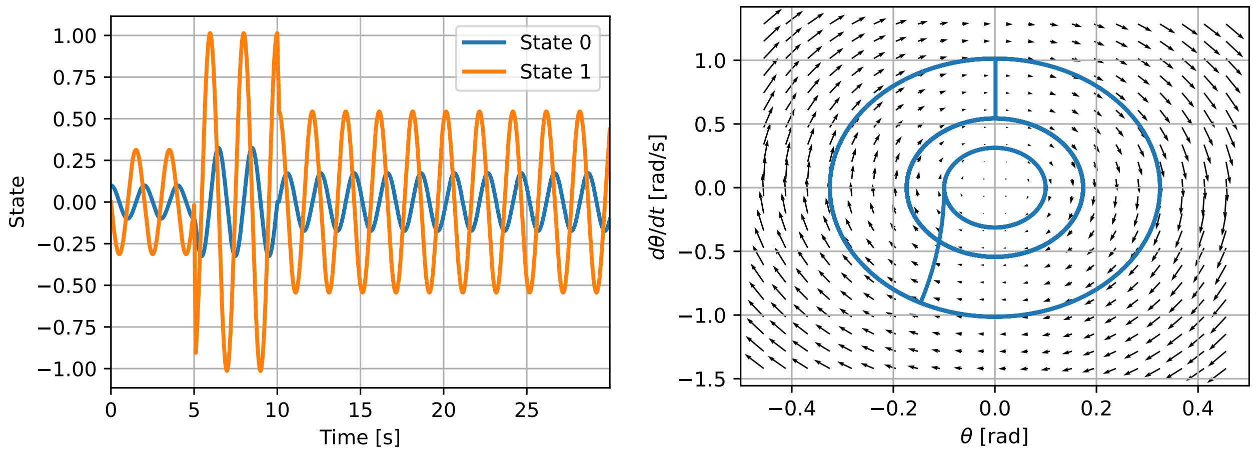
(a) Time evolution



(b) Phase portrait

Figure 7: Periodic pendulum setup (θ_0 = 0.1 [rad], $\dot{\theta}_0$ = 0 [rad/s]). The figures show that without friction the pendulum keeps oscillating without reduction of amplitude of oscillation. The fixed points, both unstable and stable, have not changed.

3.d Does the pendulum without friction (d=0) produce stable limit cycles? Discuss, and try to support your statement with some numerical simulations (show figures) and/or analytical arguments.

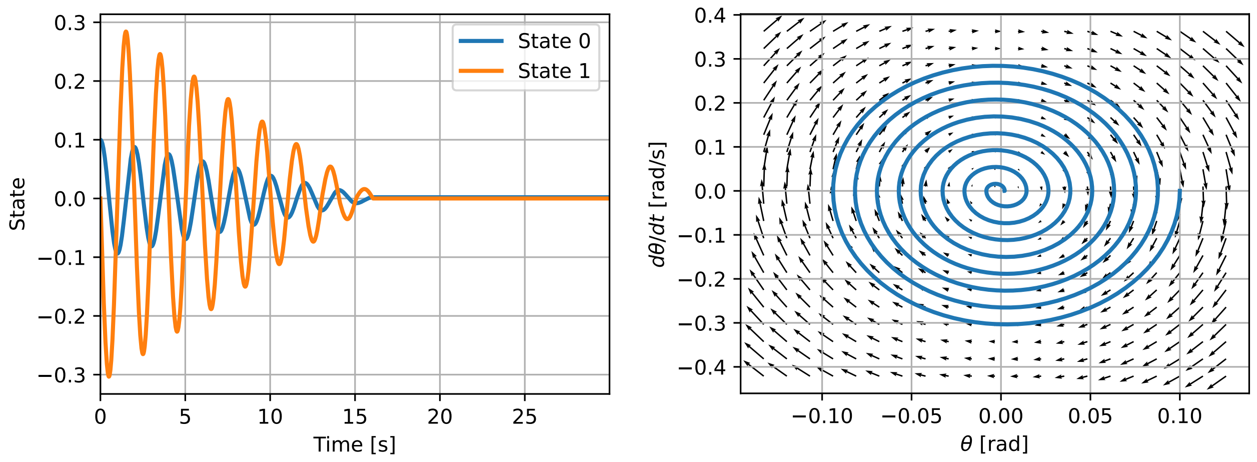


(a) Time evolution *(b) Phase portrait*

Figure 8: The pendulum without friction does not produce stable limit cycle behavior. It has multiple closed orbits that are not isolated. Perturbations lead to new orbits.

3.e Investigate how the behavior of the pendulum changes if the viscous friction term is replaced with a dry (Coulomb) friction term. Unlike viscous friction, dry friction does not depend on speed, only the direction of movement. What are the main differences between the two types of pendulum? (discuss and show some examples). And is there anything notable about the numerical integration of the pendulum with dry friction? If yes, what and why?

$$\ddot{\theta} = -\frac{g}{L}\sin\theta - d \cdot sign(\dot{\theta}) \tag{4}$$



(a) Time evolution *(b) Phase portrait*

Figure 9: Pendulum with dry friction

This represents a stiff dynamical system, i.e. a system that is difficult to integrate numerically. Indeed, this pendulum is difficult to solve numerically because of the switch of derivatives when the angular velocity changes sign (friction jumping between d and –d). Many more integration times steps are needed than for the pendulum with viscous friction. The higher d, the more difficulties the integration method has.

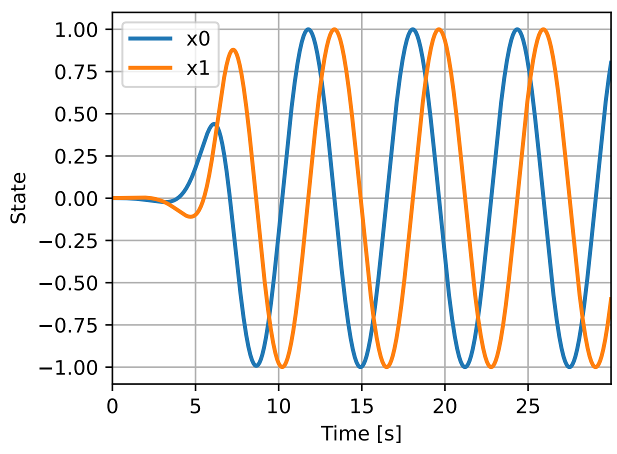
Fixed points are different from the pendulum with viscous friction. Quite complex behavior: the pendulum ends in a position/angle that depends on the initial conditions (not anymore with a vertical position).

Question 4: Coupled Hopf oscillators

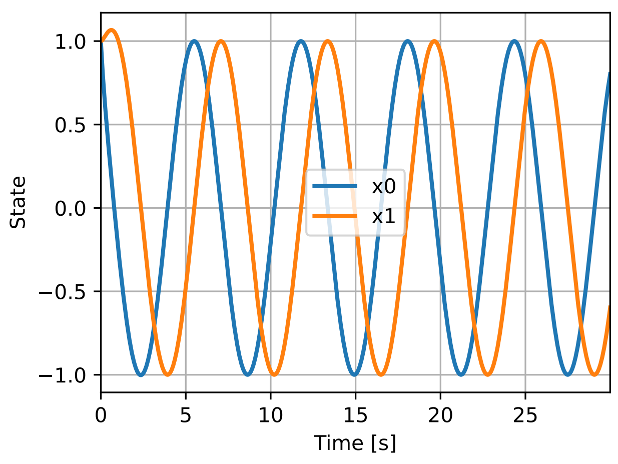
4.a Implement a single Hopf oscillator and illustrate its various behaviors using time evolution and phase plane figures. See `exercise4.py` and `ex4_hopf.py` for help with implementation.

Python code:

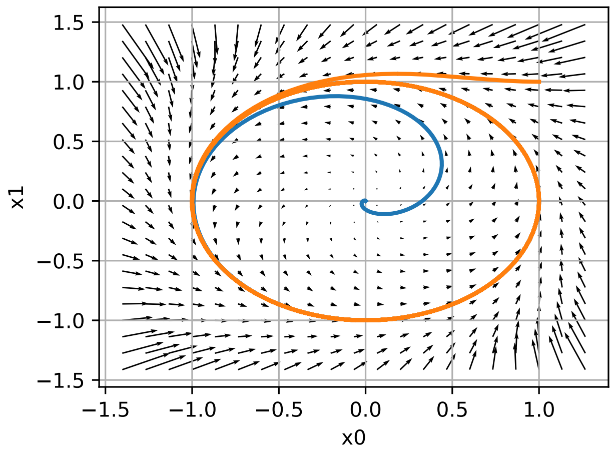
```
dx[0] = (mu - (x[0]**2 + x[1]**2)) * x[0] - omega * x[1]
dx[1] = (mu - (x[0]**2 + x[1]**2)) * x[1] + omega * x[0]
```



(a) Time evolution of case 1 (x0 = [1e-3, 1e-3])



(b) Time evolution of case 2 (x0 = [1, 1])



(c) Phase portraits of case 1 and case 2

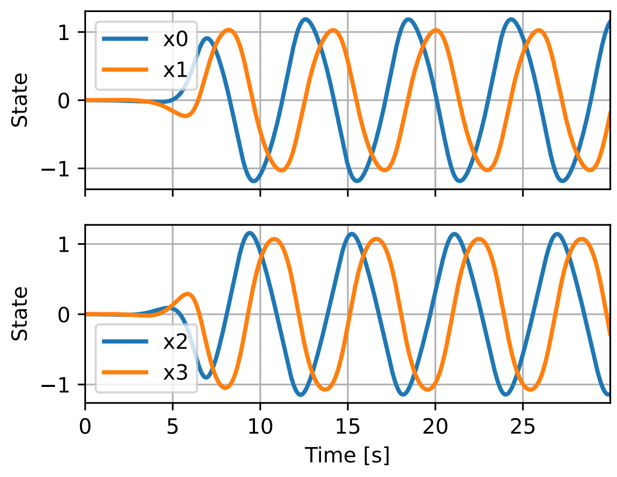
Figure 10: Hopf oscillator. As seen in the lectures, the Hopf oscillator exhibits a limit cycle with typical sine like oscillations. It has an unstable fixed point at (0,0). If mu is negative, that fixed point becomes stable and the limit cycle disappears (figures not included, but easy to make).

4.b Implement a system of two coupled Hopf oscillators and illustrate its behavior using figures showing time evolutions of states and of angles. You are free to propose your own coupling. Investigate phase locked behavior, and more generally how coupling weights and different parameters such as intrinsic frequency affect the global behavior.

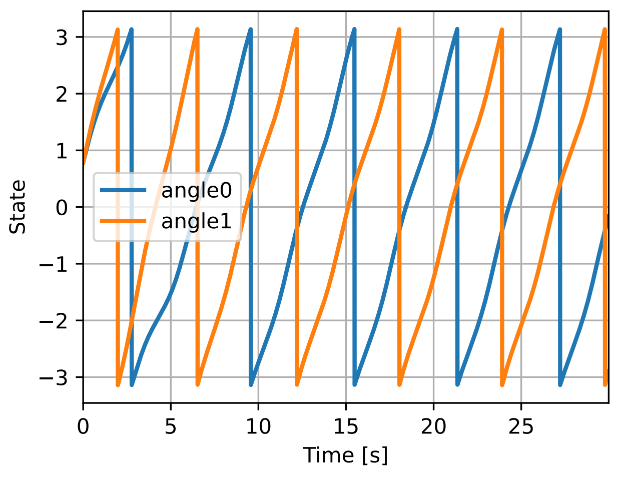
Things that should be demonstrated: phase locked regimes, regimes with drift because coupling is not strong enough compared to difference of intrinsic frequencies, difference between unidirectional versus bidirectional coupling....

Python code:

```
dx[0] = (mu[0] - (x[0]**2 + x[1]**2))*x[0] - omega[0]*x[1] + k[0]*x[2]
dx[1] = (mu[0] - (x[0]**2 + x[1]**2))*x[1] + omega[0]*x[0]
dx[2] = (mu[1] - (x[2]**2 + x[3]**2))*x[2] - omega[1]*x[3] + k[1]*x[0]
dx[3] = (mu[1] - (x[3]**2 + x[2]**2))*x[3] + omega[1]*x[2]
```

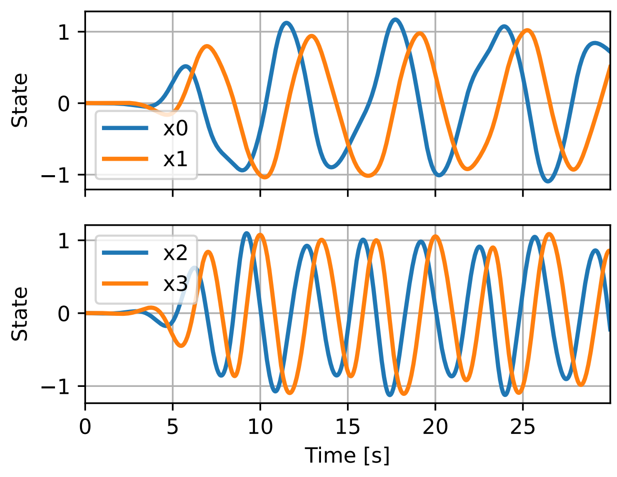


(a) Time evolution

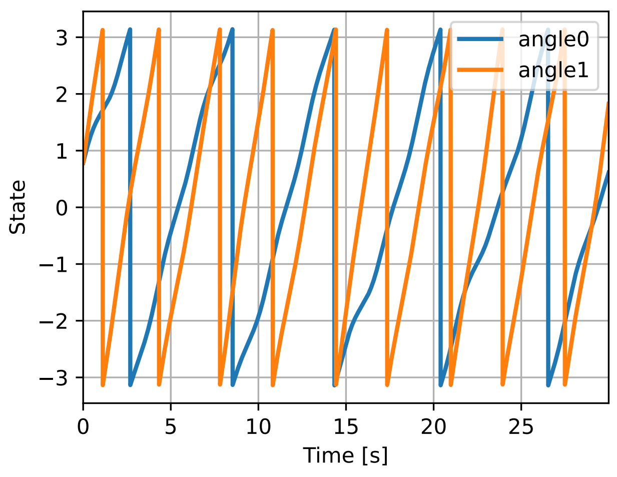


(b) Phase portrait

Figure 11: Coupled Hopf oscillator showing synchronization (Coupling parameters: $\omega_0 = 1.0$ [rad/s]; $\omega_1 = 1.2$ [rad/s] ; $K_{10} = -0.5$; $K_{01} = -0.5$, $\mu_0 = 1.0$, $\mu_1 = 1.0$). Synchronized behavior despite different intrinsic frequencies because the coupling is strong enough.



(a) Time evolution



(b) Phase portrait

Figure 12: Coupled Hopf oscillator showing drift (Coupling parameters: $\omega_0 = 1.0$ [rad/s]; $\omega_1 = 2.0$ [rad/s] ; $K_{10} = -0.5$; $K_{01} = -0.5$, $\mu_0 = 1.0$, $\mu_1 = 1.0$). The second oscillator is too fast (ω_1), and the coupling is too weak. The two oscillators drift.