hw1

September 10, 2025

1 Homework 1

due 09/10/2025 23:59

[28]:
```python
# deps
from typing import Callable
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

from utils import phase_portrait
```

1.1 Problem 3

1) Find equilibrium points for

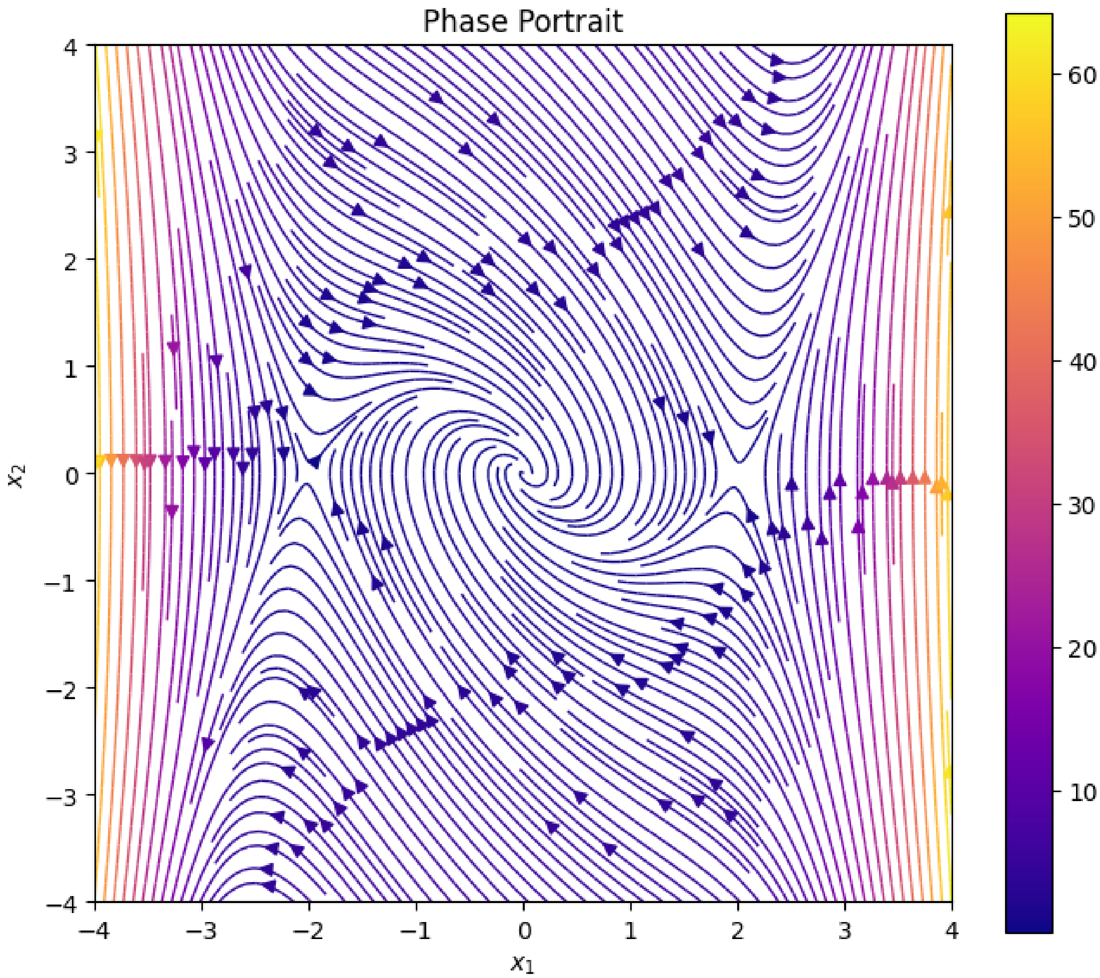
$$\dot{x}_1 = x_2 \tag{1}$$

$$\dot{x}_2 = -x_1 + \frac{1}{16}x_1^5 - x_2 \tag{2}$$

2) Plot the phase portrait

[29]:
```python
def dynamics(
    t: float | None,
    state: list,
) -> list:
    x1, x2 = state
    dx1 = x2
    dx2 = -x1 + x1**5/16 - x2
    return [dx1, dx2]

phase_portrait(x_max=4, n_pts=100, dynamics=dynamics, save=True)
```



1.2 Problem 4

1) Sketch the phase portrait for

$$\dot{x}_1 = x_2 \tag{3}$$
$$\dot{x}_2 = -kx_1 - cx_2 - \eta(x_1, x_2) \tag{4}$$

2) Describe the behavior qualitatively

   The phase portrait has an unstable node at the origin, with circular orbits on either side of the x_1 plane. The behavior is disjoint about x_2 given the sign function in the dynamics.

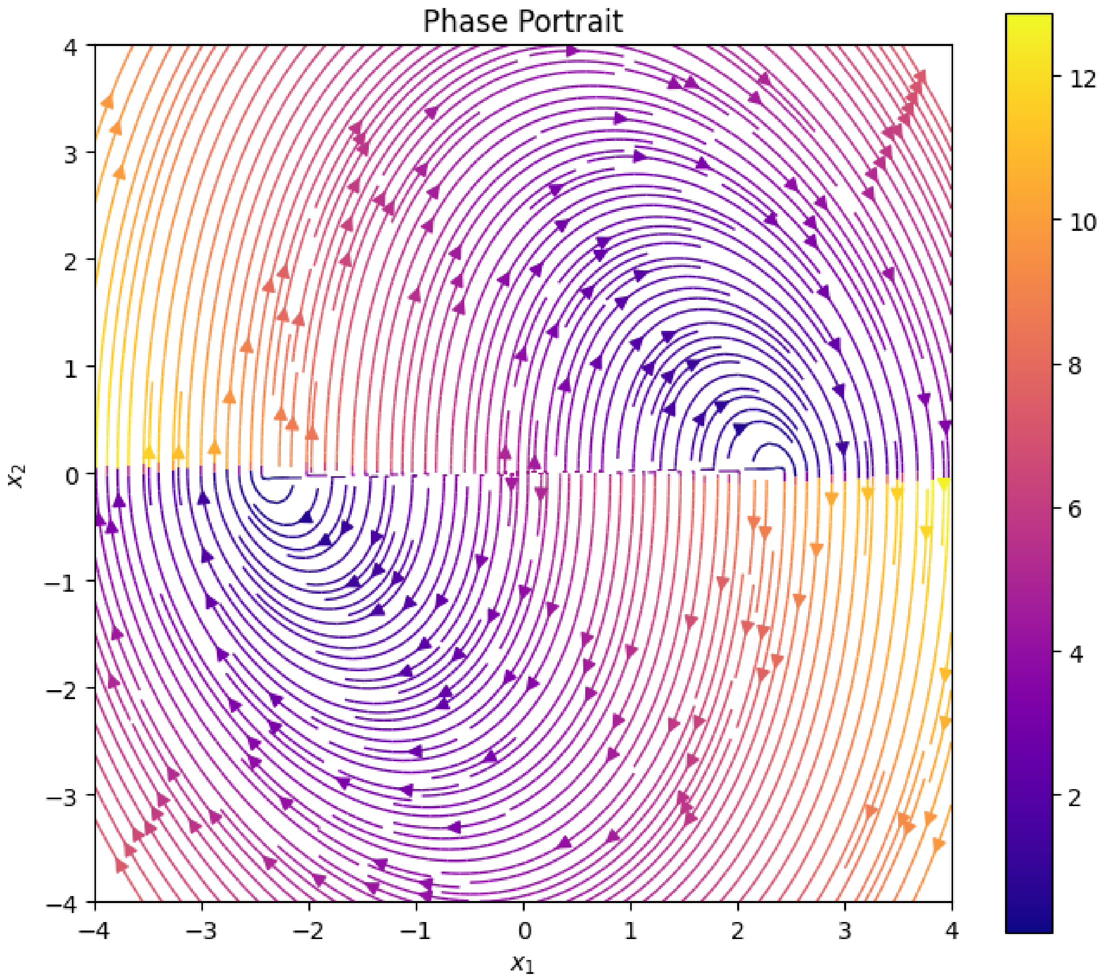
[30]:
```
def sign(n):
    return np.where(
        n > 0,
        1,
        np.where(
```

```
            n < 0,
            -1,
            0
        )
    )

def eta(
    state: list,
    k: float,
    m: float,
    g: float,
    mu_k: float,
    mu_s: float,
):
    x1, x2 = state
    return np.where(
        np.abs(x2) > 0,
        mu_k * m * g * sign(x2),
        np.where(
            np.abs(x1) <= mu_s * m * g / k,
            -k * x1,
            mu_s * m * g * sign(x1)
        )
    )

def dynamics(
    t: float | None,
    state: list,
    k: float = 2.0,
    c: float = 1.0,
    m: float = 1.0,
    g: float = -9.8,
    mu_k: float = 0.5,
    mu_s: float = 0.5,
) -> list:
    x1, x2 = state
    dx1 = x2
    dx2 = -k * x1 -c * x2 - eta(state, k, m, g, mu_k, mu_s)
    return [dx1, dx2]

phase_portrait(x_max=4, n_pts=100, dynamics=dynamics, save=True)
```



1.3 Problem 5

1) Sketch phase portraits for $u = 1$ and $u = -1$

$$\dot{x}_1 = x_2 \tag{5}$$
$$\dot{x}_2 = u \tag{6}$$

2) Develop graphical control strategy

   Define switching curve $x_1 = \pm\frac{x_2^2}{2}$. Then, from anywhere in $x_1 - x_2$ plane, the origin can be obtained by following the corresponding curve. When above the curve, apply negative control input; when below the curve, apply positive control input. This is a "bang bang" controller.

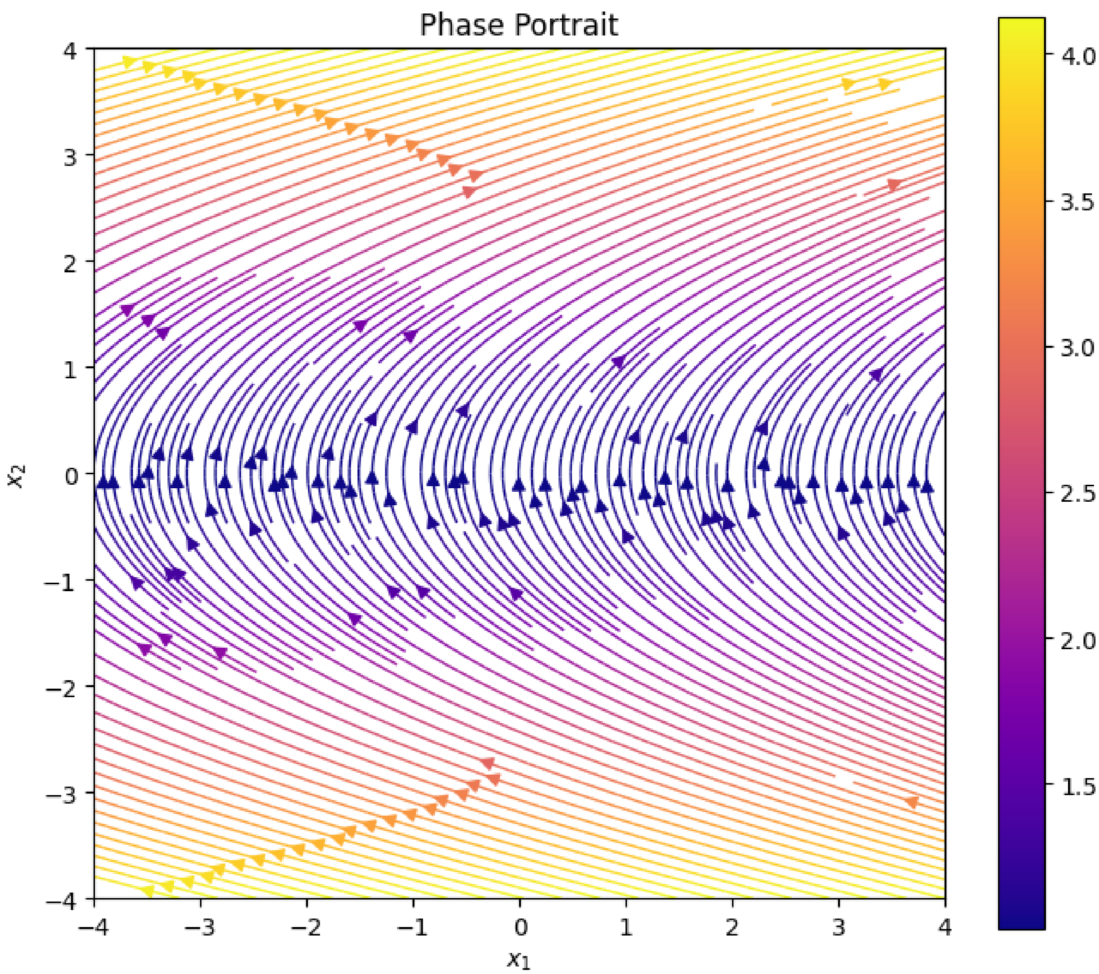
[31]:
```
def dynamics(
        t: float | None,
        state: list,
        u: float,
```

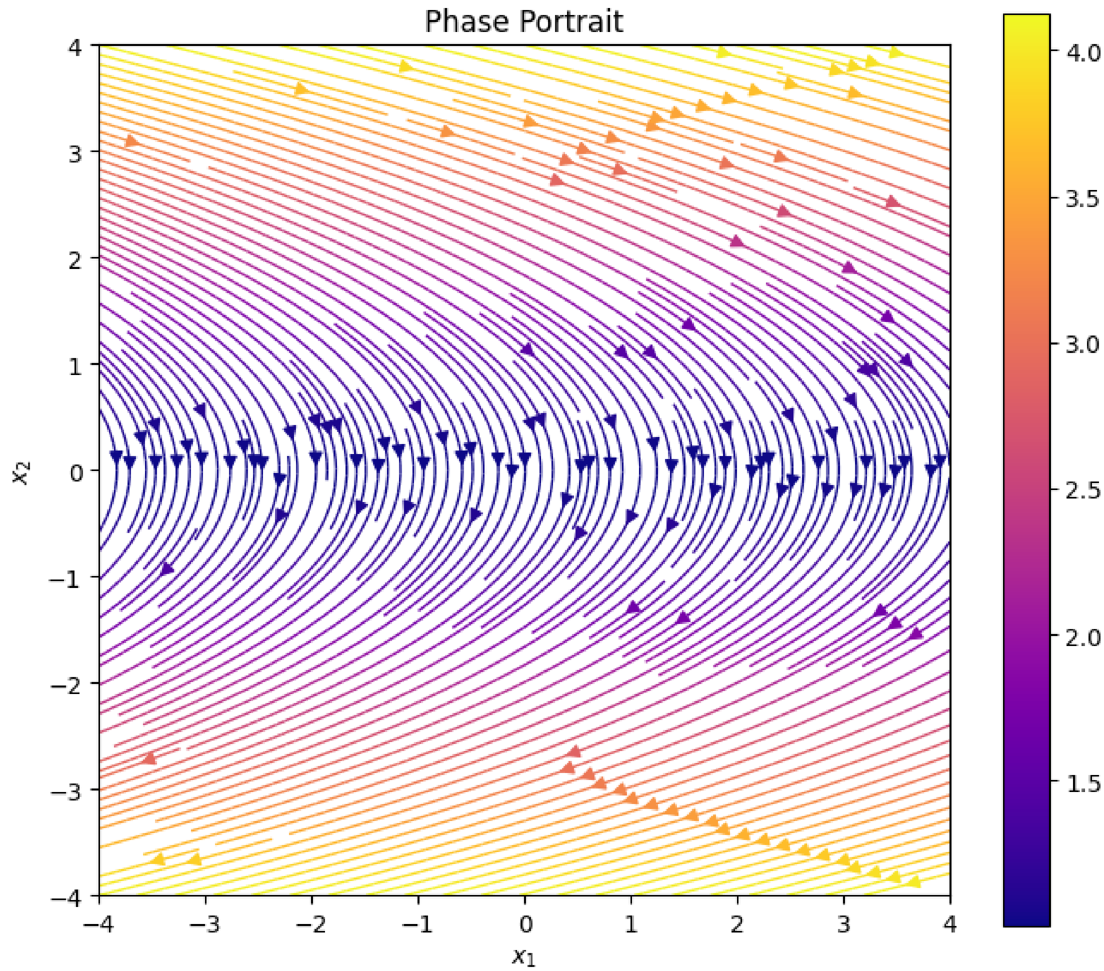
```
) -> list:
    x1, x2 = state
    dx1 = x2
    dx2 = u * np.ones_like(x2)
    return [dx1, dx2]

def set_u(u) -> Callable[..., list]:
    def wrapped_dynamics(t, state):# -> list:
        return dynamics(t, state, u)
    return wrapped_dynamics

dynamics_plus = set_u(1)
dynamics_minus = set_u(-1)

phase_portrait(x_max=4, n_pts=100, dynamics=dynamics_plus, save=True)
phase_portrait(x_max=4, n_pts=100, dynamics=dynamics_minus, save=True)
```





Phase Portrait
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4
x_1
x_2
4.0
3.5
3.0
2.5
2.0
1.5