Assignment

Damped Harmonic Oszillator

Name TN 1: Tim Peinkofer

tim.peinkofer@student.uni-tuebingen.de

Name TN 2: Fabian Kostow

fabian.kostow@student.uni-tuebingen.de

Tutor: Jose Carlos Olvera Meneses

Date: 23. Januar 2025

Inhaltsverzeichnis

1 Question 1 and 2

In the following task we want to calculate and plot the solution and it's trajectories. The code is at follows:

```
import numpy as np
import matplotlib.pyplot as plt

# Assumed Constants
k = 1  # Spring constant
m = 1  # Mass


def func(t):
    return 0 * t  # Zero function for comparison

def damped_harm_osc(t, state):
    x, y = state
    x_dot = y
    y_dot = -k / m * x - b * y
    return np.array([x_dot, y_dot])

# 4th Order Runge-Kutta
def runge_kutta(f, y_0, t_start, t_end, n):
    h = (t_end - t_start) / n  # Step size
    t_values = np.linspace(t_start, t_end, n + 1)
    y_values = np.zeros((n + 1, len(y_0)))
    y_values[0] = y_0

    for i in range(n):
        t_i = t_values[i]
        y_i = y_values[i]

        k1 = h * f(t_i, y_i)
        k2 = h * f(t_i + 0.5 * h, y_i + 0.5 * k1)
        k3 = h * f(t_i + 0.5 * h, y_i + 0.5 * k2)
        k4 = h * f(t_i + h, y_i + k3)

        y_values[i + 1] = y_i + (1 / 6) * (k1 + 2 * k2 + 2 *
                                           k3 + k4)

    return t_values, y_values
```

```
# Initial conditions
x_0 = 1  # Initial position
v_0 = 0  # Initial velocity
y_0 = [x_0, v_0]

# Time parameters
t_start = 0
t_end = 20
n = 500

for b in [0.1, 0.5, 1, 2]:
    # Solve the system
    t, result = runge_kutta(damped_harm_osc, y_0, t_start,
                                    t_end, n)

    # Extract positions and velocities
    x = result[:, 0]
    x_dot = result[:, 1]

    # Plot
    plt.figure(figsize=(10, 6))
    plt.plot(t, x, label="Position x(t)", color="blue")
    plt.plot(t, x_dot, label="Velocity x'(t)", color="orange"
                                    )
    plt.plot(t, func(t), label="Reference (x=0)", linestyle="
                                    --", color="green")
    plt.legend()
    plt.title("Damped Harmonic Oscillator")
    plt.xlabel("Time t (s)")
    plt.ylabel("Amplitude")
    plt.grid()
    plt.show()

    # Phase diagram
    plt.figure(figsize=(8, 6))
    plt.plot(x, x_dot, label="Phase trajectory", color="blue"
                                    )
    plt.title("Phase Diagram: Damped Harmonic Oscillator")
    plt.xlabel("Position x (m)")
    plt.ylabel("Velocity x'(m/s)")
    plt.axhline(0, color="black", linestyle="--", linewidth=0
                                    .8)
    plt.axvline(0, color="black", linestyle="--", linewidth=0
                                    .8)
    plt.grid()
    plt.legend()
    plt.show()
```

We have calculated the solution for the different damping cases:

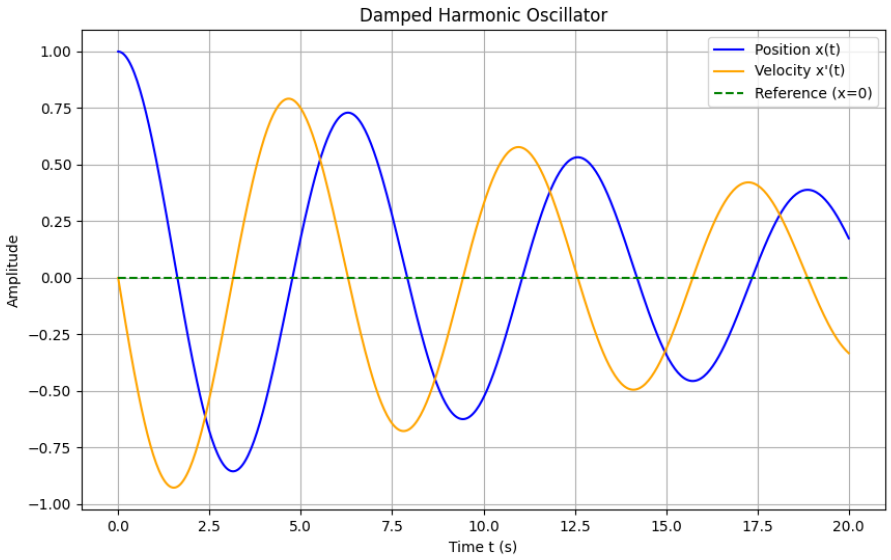

Abbildung 1: $x(t)$ with $b = 0.1$

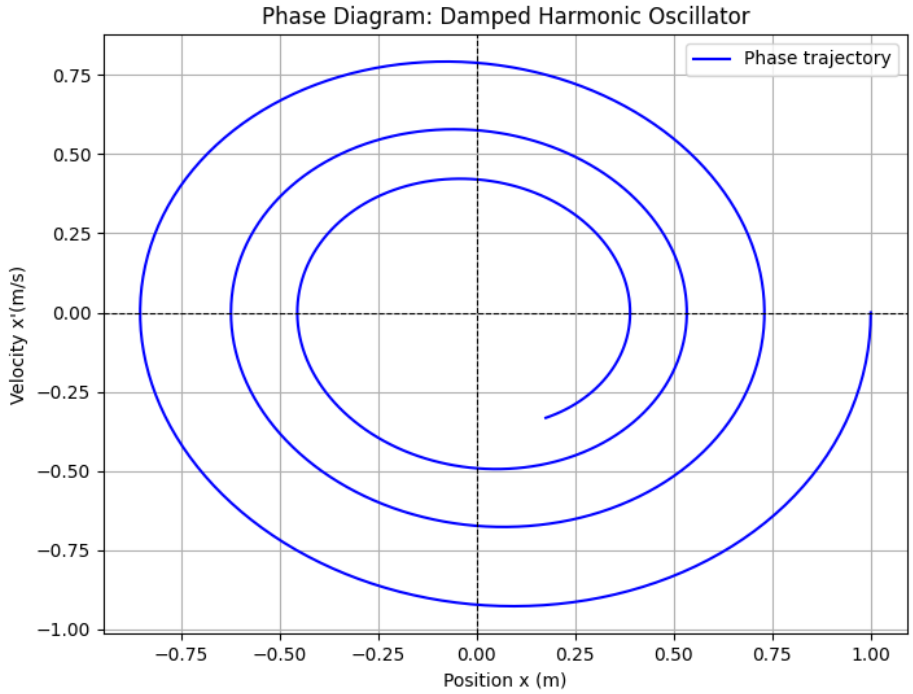

Abbildung 2: $x(t)$ with $b = 0.1$ - Phase

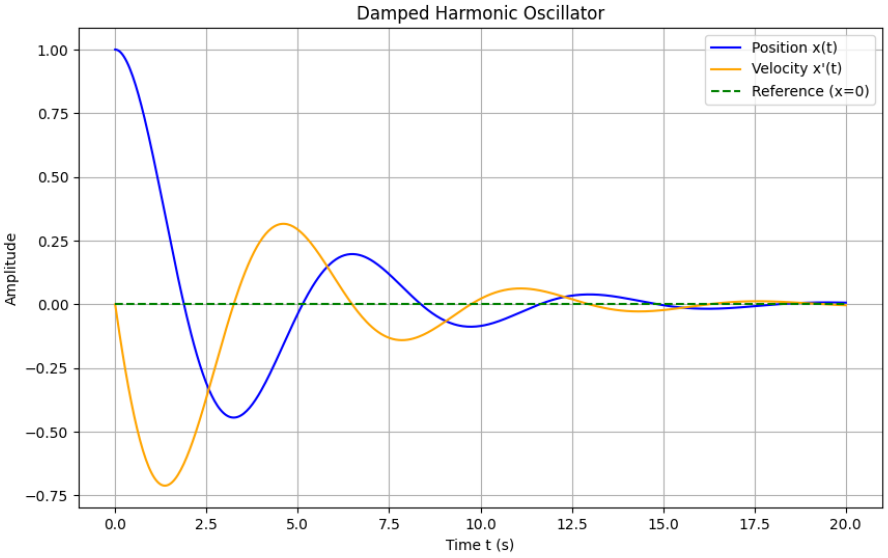

Abbildung 3: $x(t)$ with $b = 0.5$

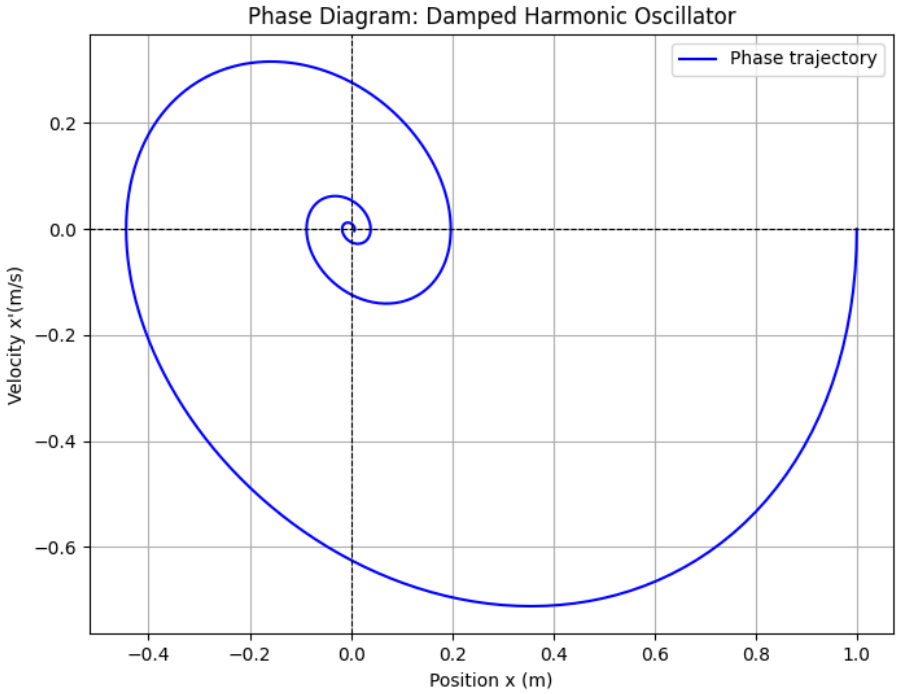

Abbildung 4: $x(t)$ with $b = 0.5$ - Phase

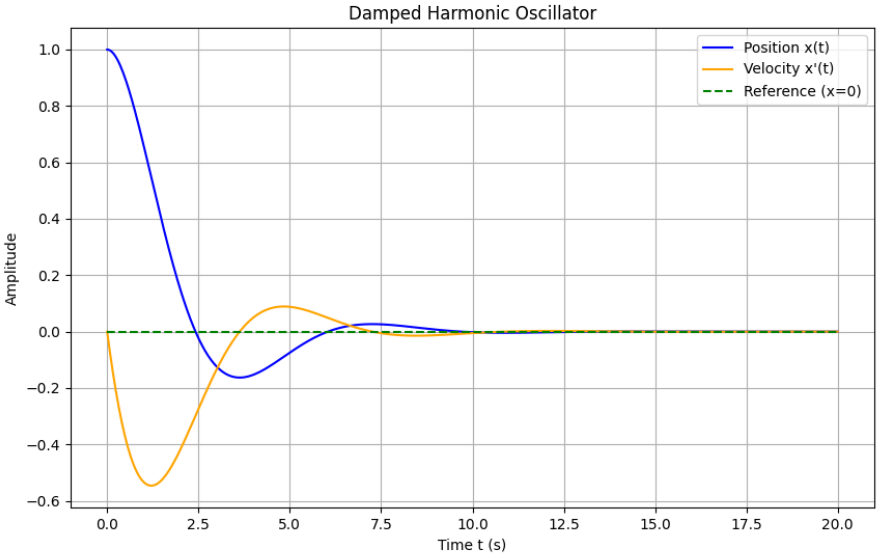

Abbildung 5: $x(t)$ with $b = 1$

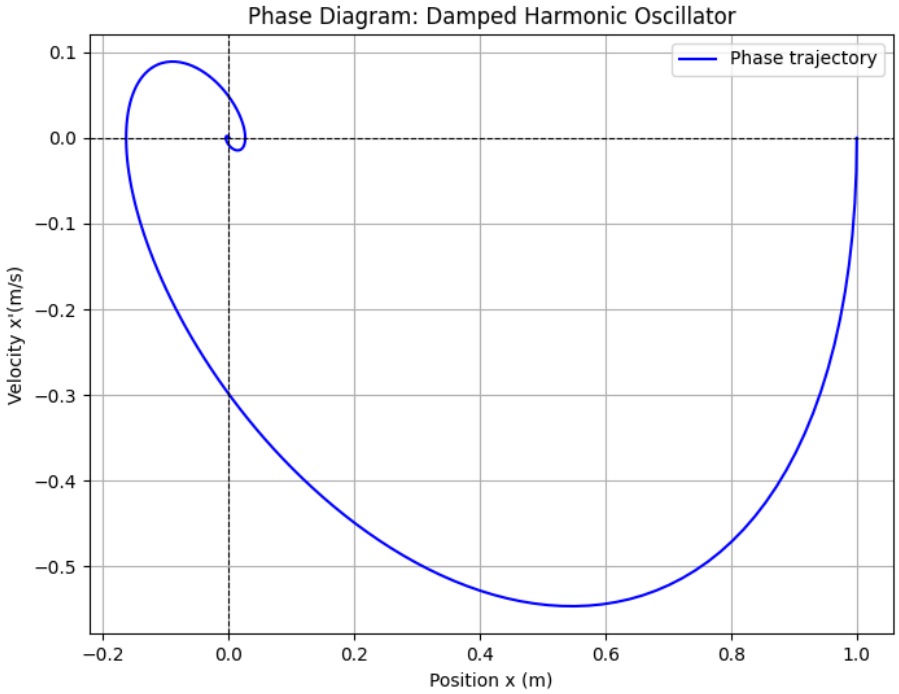

Abbildung 6: $x(t)$ with $b = 1$ - Phase

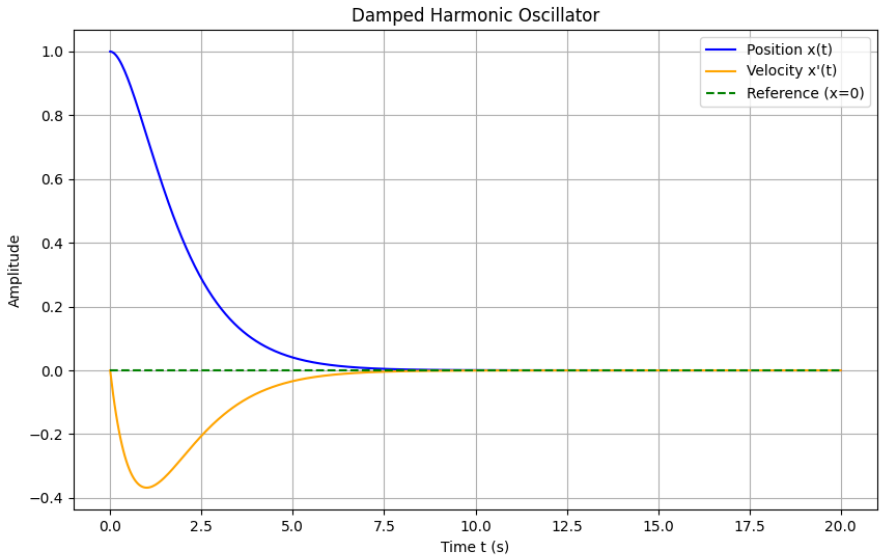

Abbildung 7: $x(t)$ with $b = 2$

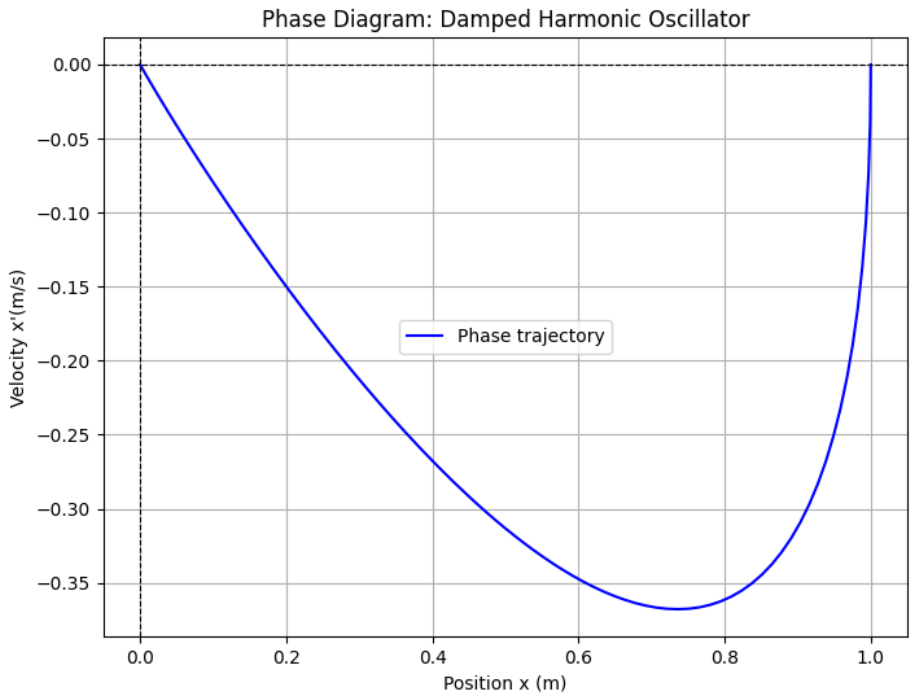

Abbildung 8: $x(t)$ with $b = 2$ - Phase

If we compare these phase diagrams with those from the previous pendulum exercise, we can see that the trajectories of the damped harmonic oscillator are not closed. This is because some of the energy is dissipated by friction. We explicitly excluded this in the previous exercise, which is why the trajectory of the previous pendulum is closed.