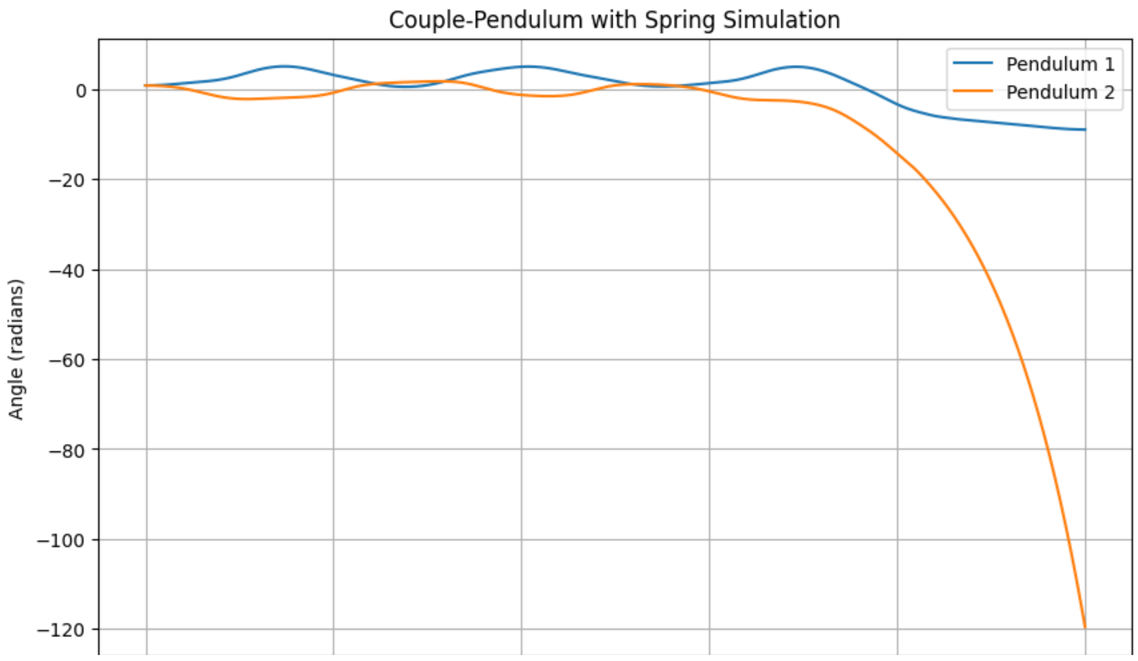
```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# System parameters
m1 = 1.0  # Mass of the first pendulum (kg)
m2 = 1.0  # Mass of the second pendulum (kg)
L1 = 1.0  # Length of the first pendulum (m)
L2 = 1.0  # Length of the second pendulum (m)
k = 1.0   # Spring constant (N/m)
g = 9.81  # Acceleration due to gravity (m/s^2)
d = 1.0   # Distance of the spring attachment from the origin (m)

# Initial conditions
theta1_0 = np.pi / 4.0  # Initial angle of the first pendulum (radians)
theta2_0 = np.pi / 4.0  # Initial angle of the second pendulum (radians)
theta1_dot_0 = 0.0      # Initial angular velocity of the first pendulum (radians/s)
theta2_dot_0 = 0.0      # Initial angular velocity of the second pendulum (radians/s)

# Define the system dynamics
def double_pendulum(t, state):
    theta1, theta2, theta1_dot, theta2_dot = state

    theta1_double_dot = (m1 * g * L1 * np.sin(theta1) - k * (theta1 - d) - m2 * g * L1 * np.sin(theta1 - theta2)) / (m1 * L1**2)
    theta2_double_dot = (m2 * g * L2 * np.sin(theta2 - theta1) - k * (d - theta2) - m2 * g * L2 * np.sin(theta2)) / (m2 * L2**2)

    return [theta1_dot, theta2_dot, theta1_double_dot, theta2_double_dot]

# Time span for the simulation
t_span = (0, 10)
t_eval = np.linspace(*t_span, 1000)

# Initial state
initial_state = [theta1_0, theta2_0, theta1_dot_0, theta2_dot_0]

# Solve the initial value problem
solution = solve_ivp(double_pendulum, t_span, initial_state, t_eval=t_eval)

# Extract the results
theta1_values = solution.y[0]
theta2_values = solution.y[1]

# Plot the motion of the double pendulum
plt.figure(figsize=(10, 6))
plt.plot(t_eval, theta1_values, label='Pendulum 1')
plt.plot(t_eval, theta2_values, label='Pendulum 2')
plt.xlabel('Time (s)')
plt.ylabel('Angle (radians)')
plt.legend()
plt.title('Couple-Pendulum with Spring Simulation')
plt.grid()
plt.show()
```


Couple-Pendulum with Spring Simulation
Angle (radians)
0
−20
−40
−60
−80
−100
−120
Pendulum 1
Pendulum 2