Experiment: Dynamics of a Damped Harmonic Oscillator

Aim

To develop a Python program using `solve_ivp` to solve and visualize the dynamics of a damped harmonic oscillator (DSHO), and to study the effect of damping on displacement, velocity, phase portrait.

Apparatus Required

Computer with Python 3 and the libraries `NumPy`, `Matplotlib`, and `SciPy` installed.

Theory

A mass–spring–dashpot system follows

$$m\,\ddot{x} + c\,\dot{x} + k\,x = 0,$$

where m is the mass, k the spring constant, and c the viscous damping coefficient. Define the natural frequency and damping ratio

$$\omega_n = \sqrt{\frac{k}{m}}, \qquad \zeta = \frac{c}{2\sqrt{km}}.$$

Behavior:

- $\zeta = 0$: Undamped SHO (pure sinusoid).
- $0 < \zeta < 1$: *Underdamped* (decaying oscillation).
- $\zeta = 1$: *Critically damped* (fastest non-oscillatory return).
- $\zeta > 1$: *Overdamped* (slow non-oscillatory return).

For the underdamped case, the solution is

$$x(t) = e^{-\zeta\omega_n t}\Big(C_1 \cos\omega_d t + C_2 \sin\omega_d t\Big), \qquad \omega_d = \omega_n\sqrt{1-\zeta^2}.$$

The total mechanical energy $E(t) = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2$ decays monotonically when $c > 0$.

For numerical integration, write the first-order system

$$\dot{x} = v, \qquad \dot{v} = -\frac{c}{m}v - \frac{k}{m}x,$$

and solve with `solve_ivp`.

Python Program (Damped SHO)

```
'' '' ''
Damped Harmonic Oscillator
--------------------------
m x'' + c x' + k x = 0  ->  x' = v,  v' = -(c/m) v - (k/m) x

Plots:
   x(t)   phase portrait (v vs x)
'' '' ''

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

# Parameters (edit here)
m = 1.0       # mass
k = 4.0       # stiffness
c = 0.6       # damping coefficient (set 0 for undamped)
x0 = 1.0      # initial displacement
v0 = 0.0      # initial velocity
t_span = (0.0, 20.0)
t_eval = np.linspace(*t_span, 3000)

# Derived quantities
omega_n = np.sqrt(k/m)
zeta = c/(2*np.sqrt(k*m))
print(f''omega_n = {omega_n:.4f} rad/s,  zeta = {zeta:.4f}'')

# ODE system: y = [x, v]
def dsho(t, y):
    x, v = y
    return [v, -(c/m)*v - (k/m)*x]

# Solve
sol = solve_ivp(dsho, t_span, [x0, v0], t_eval=t_eval, rtol=1e-9,
    atol=1e-12)
t = sol.t
x = sol.y[0]
v = sol.y[1]

# Energies
T = 0.5*m*v**2
U = 0.5*k*x**2
E = T + U

# Plots
plt.figure(figsize=(7,4))
plt.plot(t, x, lw=1.6)
plt.xlabel('t (s)'); plt.ylabel('x(t)')
plt.title('Displacement vs Time')
```

```python
plt.grid(True); plt.tight_layout()

plt.figure(figsize=(7,4))
plt.plot(t, v, lw=1.6)
plt.xlabel('t (s)'); plt.ylabel('v(t)')
plt.title('Velocity vs Time')
plt.grid(True); plt.tight_layout()

plt.figure(figsize=(6,6))
plt.plot(x, v, lw=1.6)
plt.xlabel('x'); plt.ylabel('v')
plt.title('Phase Portrait (v vs x)')
plt.grid(True); plt.axis('equal'); plt.tight_layout()

plt.figure(figsize=(7,4))
plt.plot(t, E, lw=1.8, label='Total E')
plt.plot(t, T, lw=1.0, ls='--', label='Kinetic T')
plt.plot(t, U, lw=1.0, ls='--', label='Potential U')
plt.xlabel('t (s)'); plt.ylabel('Energy')
plt.title('Energy vs Time')
plt.grid(True); plt.legend(); plt.tight_layout()

plt.show()
```

Listing 1: Damped harmonic oscillator solved with `solve_ivp`.

Procedure

1. Specify m, k, c and initial conditions (x_0, v_0).
2. Formulate the first-order system and integrate with `solve_ivp`.
3. Plot $x(t)$ and the phase portrait (v vs x).
4. Vary c to observe underdamped, critically damped, and overdamped regimes.

Observations and Results

With $m = 1$, $k = 4$ ($\omega_n = 2$ rad/s) and $c = 0.6$,

$$\zeta = \frac{c}{2\sqrt{km}} = \frac{0.6}{4} = 0.15 \quad \text{(underdamped)}.$$

The displacement and velocity exhibit decaying oscillations; the phase portrait spirals into the origin. The total energy decays monotonically due to damping.

Conclusion

The numerical solution confirms the expected behavior of a damped harmonic oscillator: for $0 < \zeta < 1$ the motion is an exponentially decaying sinusoid, the phase trajectory is a spiral, and total mechanical energy decreases with time.

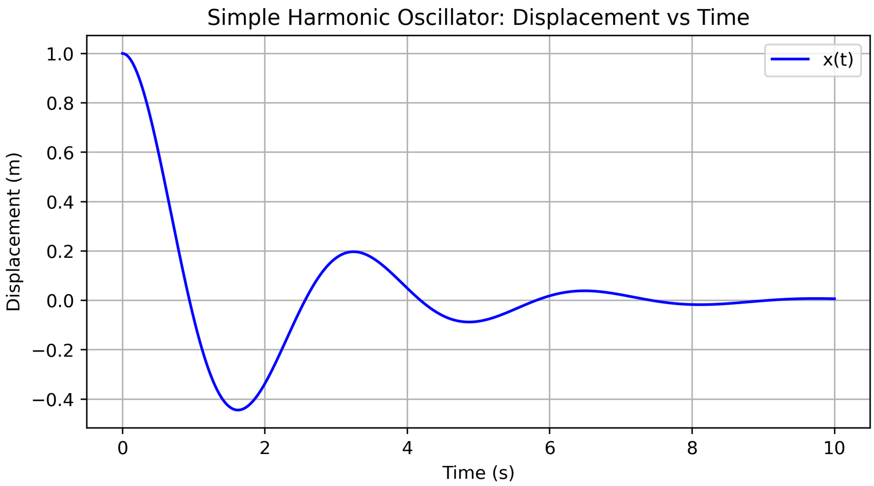

Figure 1: Displacement $x(t)$ vs time for damped SHO ($\zeta = 0.15$).

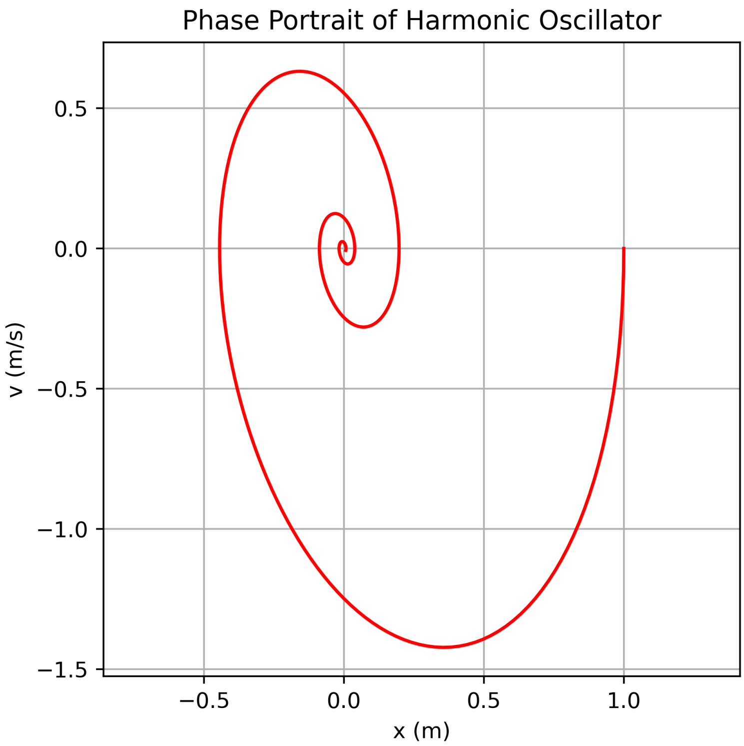

Figure 2: Phase portrait (v vs x) showing inward spiral.

Precautions

1. Choose Δt (via dense `t_eval`) fine enough to resolve oscillations.
2. Verify parameters give the intended regime ($\zeta < 1, = 1, > 1$).
3. In the undamped limit ($c \to 0$), check that energy remains constant numerically.