Experiment: Dynamics of a Harmonic Oscillator

Aim

To develop a Python program for solving and visualizing the dynamics of a simple harmonic oscillator using the numerical integration method.

Apparatus Required

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

Theory

A harmonic oscillator consists of a mass m attached to a spring of stiffness constant k. The restoring force is given by Hooke's law:

$$F = -kx.$$

Applying Newton's second law,

$$m\frac{d^2x}{dt^2} = -kx,$$

or

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0.$$

Let $\omega = \sqrt{k/m}$ be the angular frequency of oscillation. The analytical solution is:

$$x(t) = A\cos(\omega t) + B\sin(\omega t),$$

where A and B depend on initial displacement and velocity.

For numerical computation, the equation is written as a first-order system:

$$\begin{cases} \dot{x} = v, \\ \dot{v} = -\dfrac{k}{m}x. \end{cases}$$

The system is solved using the built-in `solve_ivp` function in `SciPy`, which implements adaptive Runge–Kutta methods.

Python Program

```

Simple Harmonic Oscillator
--------------------------
Equation:  m x'' + k x = 0
           => x'' = -(k/m) * x

```

```python
This program solves the equation using solve_ivp
and plots displacement and phase trajectory.


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

# Parameters
m = 1.0          # mass (kg)
k = 4.0          # spring constant (N/m)
x0 = 1.0         # initial displacement (m)
v0 = 0.0         # initial velocity (m/s)
t_span = (0, 10)     # time interval
t_eval = np.linspace(*t_span, 500)    # points for evaluation

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

# Solve ODE
sol = solve_ivp(SHO, t_span, [x0, v0], t_eval=t_eval)

# Extract results
t = sol.t
x = sol.y[0]
v = sol.y[1]

# Plot displacement vs time
plt.figure(figsize=(7,4))
plt.plot(t, x, label='x(t)', color='blue')
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Simple Harmonic Oscillator: Displacement vs Time')
plt.grid(True)
plt.legend()
plt.tight_layout()

# Plot phase trajectory
plt.figure(figsize=(5,5))
plt.plot(x, v, color='red')
plt.xlabel('x (m)')
plt.ylabel('v (m/s)')
plt.title('Phase Portrait of Harmonic Oscillator')
plt.grid(True)
plt.axis('equal')
plt.tight_layout()

```

```
plt.show()
```

Listing 1: Simulation of a simple harmonic oscillator using solve$_{i}vp$.

Procedure

1. Write the governing differential equation $m\ddot{x} + kx = 0$.
2. Convert it into two first-order equations for x and v.
3. Implement the system using `solve_ivp` in Python.
4. Plot $x(t)$ and the phase trajectory (v vs x).
5. Observe the periodic nature of motion and verify that the trajectory is an ellipse in phase space.

Observation and Results

For the parameters $m = 1$ kg and $k = 4$ N/m, the natural angular frequency is

$$\omega = \sqrt{\frac{k}{m}} = 2 \text{ rad/s}.$$

The displacement $x(t)$ varies sinusoidally with time, and the phase plot v vs x is an ellipse, confirming simple harmonic motion.

Result

The numerical solution correctly reproduces the sinusoidal motion of the simple harmonic oscillator. The period of oscillation obtained from the graph agrees with the theoretical value:

$$T = \frac{2\pi}{\omega} = \pi \text{ seconds}.$$

Precautions

1. Ensure correct values of m and k are used.
2. Use sufficiently small time steps for accurate visualization.
3. Verify that total energy remains constant in the undamped case.

oscillator_xt.png

Figure 1: Displacement $x(t)$ vs time showing sinusoidal motion.

oscillator_phase.png

Figure 2: Phase portrait (v vs x) of the harmonic oscillator.