## 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:

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

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
```

```
This program solves the equation using solve_ivp
  and plots displacement and phase trajectory.
9
  import numpy as np
  import matplotlib.pyplot as plt
12
  from scipy.integrate import solve_ivp
13
14
  # Parameters
  m = 1.0
                 # mass (kg)
16
  k = 4.0
                 # spring constant (N/m)
17
                 # initial displacement (m)
  x0 = 1.0
18
  v0 = 0.0
                 # initial velocity (m/s)
  t_{span} = (0, 10)
                        # time interval
  t_eval = np.linspace(*t_span, 500) # points for evaluation
21
  # ODE system: y = [x, v]
23
  def SHO(t, y):
24
       x, v = y
       dxdt = v
       dvdt = -(k/m) * x
27
       return [dxdt, dvdt]
29
  # Solve ODE
30
  sol = solve_ivp(SHO, t_span, [x0, v0], t_eval=t_eval)
31
  # Extract results
33
  t = sol.t
34
  x = sol.y[0]
35
  v = sol.y[1]
36
  # Plot displacement vs time
38
  plt.figure(figsize=(7,4))
39
  plt.plot(t, x, label='x(t)', color='blue')
40
  plt.xlabel('Time (s)')
41
  plt.ylabel('Displacement (m)')
42
  plt.title('Simple Harmonic Oscillator: Displacement vs Time')
  plt.grid(True)
44
  plt.legend()
45
  plt.tight_layout()
46
47
  # Plot phase trajectory
48
  plt.figure(figsize=(5,5))
  plt.plot(x, v, color='red')
  plt.xlabel('x (m)')
51
  plt.ylabel('v (m/s)')
  plt.title('Phase Portrait of Harmonic Oscillator')
  plt.grid(True)
  plt.axis('equal')
  plt.tight_layout()
56
57
```

1

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:

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

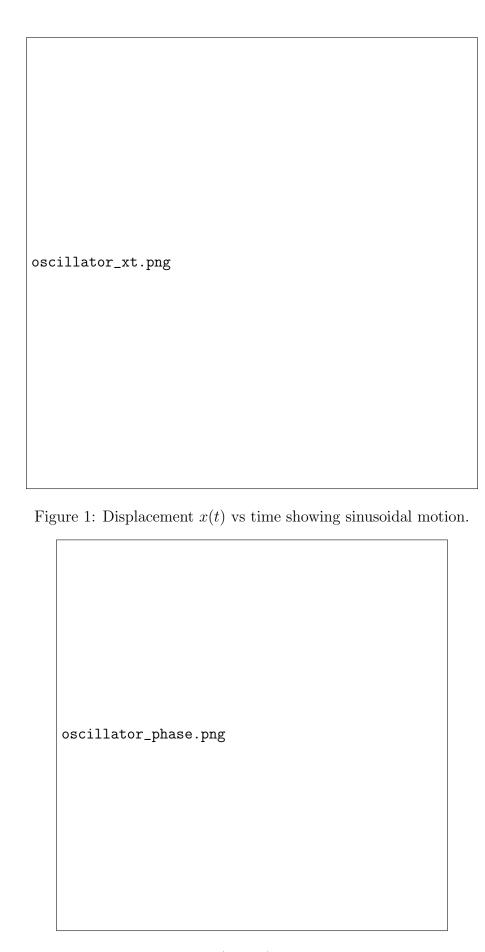


Figure 2: Phase portrait (v vs x) of the harmonic oscillator.