# PH504M Lab 2: Basic Python Coding to Solve Physics Problems

Vikram Khaire

17 Januray 2025

## 1. Projectile Motion

Simulate the trajectory of a projectile launched with an initial speed $v_0$ at an angle $\theta$ from the horizontal.

The equations of motion are:

$$x = v_0 \cos(\theta) t, \quad y = v_0 \sin(\theta) t - \frac{1}{2} g t^2$$

where $g = 9.8\ \mathrm{m/s^2}$.

**Tasks:**

1. Write a function `projectile_trajectory(v0, theta)` to calculate $x$ and $y$ positions using a `for` loop and return two lists: `x_positions` and `y_positions`.
2. Ensure the loop stops when $y \leq 0$ (projectile hits the ground) using an `if` statement.
3. Write a separate function `plot_trajectory(x, y)` to plot $x$ vs. $y$ using Matplotlib.

**Basic Plotting Code for Guidance:**

```python
import matplotlib.pyplot as plt

def plot_trajectory(x, y):
    plt.figure()
    plt.plot(x, y, label="Projectile Trajectory")
    plt.xlabel("Distance (m)")
    plt.ylabel("Height (m)")
    plt.title("Projectile Motion")
    plt.legend()
    plt.grid(True)
    plt.show()
```

## 2. Heat Distribution in a Metal Plate

Simulate heat diffusion in a $5 \times 5$ metal plate.

**Tasks:**

1. Write a function `initialize_plate()` to create a $5 \times 5$ NumPy array where boundary cells are set to 100°C, and interior cells are 0°C.
2. Write a function `simulate_heat_diffusion(plate, iterations)` to update the temperature of each interior cell using the average of its four neighbors for a given number of iterations.
3. Use Matplotlib to plot the heat distribution as a heatmap with a function `plot_heatmap(plate)`.

**Basic Plotting Code for Heatmap:**

```python
def plot_heatmap(plate):
    plt.imshow(plate, cmap="hot", interpolation="nearest")
    plt.colorbar(label="Temperature (C)")
    plt.title("Heat Distribution")
    plt.show()
```

## 3. Ideal Gas Law

The Ideal Gas Law is given by:

$$PV = nRT$$

where $P$ is pressure, $V$ is volume, $n$ is the number of moles, $R = 8.314$ J/mol·K, and $T$ is temperature.

**Tasks:**

1. Write a function `calculate_pressure(V, T, n)` to compute $P$ given $V$, $T$, and $n$. Use NumPy arrays for $V$ and $T$.
2. Write another function `plot_3d_surface(V, T, P)` to create a 3D surface plot of $P$ as a function of $V$ and $T$.

**Basic Plotting Code for 3D Surface:**

```python
from mpl_toolkits.mplot3d import Axes3D

def plot_3d_surface(V, T, P):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    V, T = np.meshgrid(V, T)
    ax.plot_surface(V, T, P, cmap="viridis")
    ax.set_xlabel("Volume (L)")
    ax.set_ylabel("Temperature (K)")
    ax.set_zlabel("Pressure (Pa)")
    ax.set_title("Ideal Gas Law Surface")
    plt.show()
```

# 4. Motion of a Mass on a Spring

The motion of a mass attached to a spring is governed by:

$$x(t) = A\cos(\omega t)$$

where $A$ is the amplitude, $\omega = \sqrt{\frac{k}{m}}$, $k$ is the spring constant, and $m$ is the mass.

**Tasks:**

1. Write a function `spring_motion(A, k, m, t_values)` to compute $x(t)$ for a given list of time values $t$ using NumPy arrays.

2. Write another function `plot_motion(t, x)` to plot $x(t)$ vs. $t$.

**Basic Plotting Code:**

```
def plot_motion(t, x):
    plt.figure()
    plt.plot(t, x, label="Spring Motion")
    plt.xlabel("Time (s)")
    plt.ylabel("Displacement (m)")
    plt.title("Mass on a Spring")
    plt.legend()
    plt.grid(True)
    plt.show()
```

# 5. Electric Field of a Point Charge

The electric field $E$ of a point charge $q$ at a distance $r$ is given by:

$$E = \frac{kq}{r^2}$$

where $k = 9 \times 10^9\ \mathrm{Nm^2/C^2}$.

**Tasks:**

1. Write a function `electric_field(q, x, y)` to calculate $E_x$ and $E_y$ components on a grid of points in the XY plane.

2. Write another function `plot_field(x, y, Ex, Ey)` to plot the electric field using a quiver plot.

**Basic Plotting Code for Quiver:**

```
def plot_field(x, y, Ex, Ey):
    plt.figure()
    plt.quiver(x, y, Ex, Ey, scale=1e11, color="blue")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    plt.title("Electric Field of a Point Charge")
    plt.grid(True)
    plt.show()
```