Portfolio Work 1

Labs and Data Analysis 2

Zakilya Watson-Jarrett

All code used for the simulations is appended in the accompanying Python file.

Question 1

The Euler method is a numerical technique to approximate solutions to ordinary differential equations (ODEs).

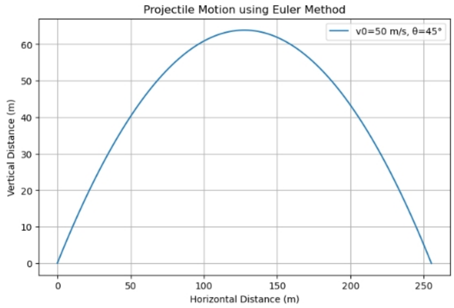

Figure 1: Projectile Motion using Euler Method

As shown in Figure 1, the projectile motion is computed and plotted using python. The code allows for easy modification of the initial speed, launch angle, and gravity. A 'while' loop is used to track movement until the projectile lands. My comments clarify key steps.

Question 2

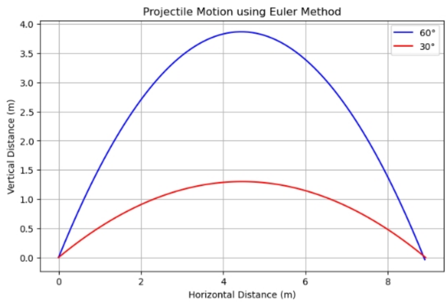

Figure 2: Projectile motion

Comparison with Theoretical Values

The theoretical range R is given by:

$$R = \frac{v_0^2 \sin(2\theta)}{g}$$

For $\theta = 60°$ and $\theta = 30°$:

$$R = \frac{10^2 \sin(120°)}{9.81} \approx 8.83 \text{ m}$$

Since $\sin(2\theta)$ is the same for both angles, the range is identical.

b) The range of a projectile are dependent on the initial velocity and the angle of projection. The graph shows that the range calculated using the projectile motion formula are consistent with the SUVAT equations. This implies the calculations are accurate

The angles 60°and 30°are complimentary as they add up to 90°. This makes the range of the projectile the same, hence the same horizontal distance in the absence of drag force.

Moreover, a larger angle of projection produces a larger vertical component of the initial velocity, leading to a greater maximum height. This explains why the projectile at 60°reaches a higher maximum height than at 30°.

Effect of Time Step (dt)

The time step affects the numerical accuracy. For example, a smaller dt (e.g. 0.001 s) is more accurate but a slower computation. However, a larger dt (e.g. 0.05 s) has a faster computation but is less accurate. To create an optimal choice a dt of 0.01 s was picked. This provides a balance between accuracy and efficiency

Question 3

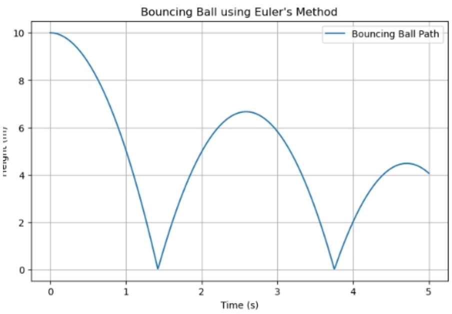

Figure 3: Bouncing Ball Simulation

The bouncing ball simulation uses a hard-sphere model, where velocity reverses upon ground collision, reduced by a coefficient of restitution (COR):

- Velocity reversal: $v_y = -v_y \times COR$
- Position correction: $y = -y$ to keep the ball above ground.

The movement of the bounce ball was simulated with an initial height of 10 m,a COR of 0.8, and a timestep of 0.01 s. The graph shows successive bounces with diminishing heights, consistent with energy loss due to the COR

Initially, a small $dt = 0.001$ was used, giving a smooth but slow simulation. However, a larger $dt = 0.1$ made the ball sink below the ground before bouncing, which is unrealistic. This can lead to instability, particularly in systems with frequent interactions such as bouncing balls. A compromise at $dt = 0.01$ was chosen, balancing precision and efficiency.