📘 **Report**

Physics-Based Simulation and Mathematical Modeling Using Python

---

1️⃣ Introduction

Physics-based simulation involves representing real-world physical systems using mathematical equations and solving them computationally. Instead of solving equations only on paper, we use programming (Python) to:

- Model physical laws
- Perform numerical calculations
- Visualize motion and system behavior
- Analyze dynamic systems over time

In this project, three different physics systems were modeled:

1. Free Fall Motion
2. Projectile Motion
3. Spring-Mass Oscillation (Differential Equation System)

2️⃣ Tools and Technologies Used

The following Python libraries were used:

- **NumPy** – For numerical calculations and array handling
- **Matplotlib** – For plotting graphs and visualizing motion
- **SciPy (odeint)** – For solving differential equations numerically

These libraries are widely used in scientific computing and engineering simulations.

3️⃣ Experiment 1: Free Fall Simulation

📌 Objective

To simulate the vertical motion of an object falling under gravity.

📌 Theoretical Background

The motion of a freely falling object is governed by the kinematic equation:

$y = y_0 + v_0t - (1/2)gt^2$

Where:

- y_0 = initial height
- v_0 = initial velocity
- g = acceleration due to gravity ($9.8\ m/s^2$)
- t = time

This equation assumes:

- No air resistance
- Constant gravitational acceleration

📌 Methodology

1. Defined gravitational acceleration ($g = 9.8\ m/s^2$).
2. Set initial height = 100 meters.
3. Set initial velocity = 0 m/s.
4. Generated time values using NumPy.
5. Applied the motion equation.
6. Plotted height vs time graph using Matplotlib.

📌 Observations

- The graph is a downward-opening parabola.
- Height decreases non-linearly over time.
- Acceleration remains constant throughout the motion.
- The object reaches ground when height becomes zero.

📌 Conclusion

The simulation successfully demonstrates uniformly accelerated motion under gravity and verifies theoretical kinematic equations through computational modeling.

4️⃣ Experiment 2: Projectile Motion Simulation

📌 Objective

To simulate the two-dimensional motion of a projectile launched at an angle.

📌 Theoretical Background

Projectile motion is described by two independent equations:

Horizontal Motion:
$x = v_0 \cos(\theta) \cdot t$

Vertical Motion:
$y = v_0 \sin(\theta) \cdot t - (1/2)gt^2$

Where:

- v_0 = initial velocity
- θ = angle of projection
- g = gravitational acceleration

Assumptions:

- No air resistance
- Constant gravity
- Independent horizontal and vertical motion

📌 Methodology

1. Set initial velocity = 20 m/s.
2. Converted launch angle (45°) into radians.
3. Generated time array.
4. Calculated horizontal (x) and vertical (y) positions.
5. Plotted y vs x graph.

📌 Observations

- The trajectory forms a parabolic path.
- Maximum height occurs at mid-flight.
- Horizontal motion remains uniform.
- Vertical motion is accelerated due to gravity.

📌 Conclusion

The simulation validates the physics principle that horizontal and vertical components of motion act independently. The graphical output confirms classical projectile motion behavior.

5️⃣ Experiment 3: Spring-Mass System (Differential Equation Model)

📌 Objective

To simulate oscillatory motion of a mass attached to a spring using numerical methods.

📌 Theoretical Background

The motion of a spring-mass system is governed by:

$m(d^2x/dt^2) + kx = 0$

Where:

- m = mass
- k = spring constant
- x = displacement

This is a second-order differential equation.

To solve it numerically, it is converted into two first-order equations:

$dx/dt = v$
$dv/dt = -(k/m)x$

This system is solved using the numerical method (odeint).

📌 Methodology

1. Defined spring constant (k = 4).
2. Defined mass (m = 1).
3. Converted second-order equation into first-order system.
4. Set initial conditions (x = 1, v = 0).
5. Used odeint to solve the system.
6. Plotted displacement vs time.

📌 Observations

- The graph shows sinusoidal oscillation.
- Motion is periodic.
- Amplitude remains constant (ideal system).
- System demonstrates simple harmonic motion.

📌 Conclusion

The spring-mass simulation demonstrates how differential equations describe real-world physical systems. Using numerical solvers allows complex systems to be analyzed computationally.

6️⃣ Comparative Analysis

Experiment	Type of Motion	Mathematical Level	Nature of System
Free Fall	1D Accelerated Motion	Algebraic Equation	Deterministic
Projectile Motion	2D Motion	Algebraic Equations	Deterministic
Spring-Mass	Oscillatory Motion	Differential Equation	Dynamic System

---

7️⃣ Overall Learning Outcomes

Through these simulations:

- Understood mathematical modeling of physical systems
- Applied physics equations computationally

- Learned numerical simulation techniques
- Visualized dynamic motion using graphs
- Solved differential equations using SciPy

8️⃣ Final Conclusion

Physics-based simulation using Python provides a powerful method to analyze and visualize real-world systems. From simple kinematic equations to complex differential equation models, computational tools allow accurate and efficient modeling of physical behavior.

These simulations demonstrate how mathematics, physics, and programming integrate to form the foundation of modern scientific computing.