FOUR-YEAR UNDER GRADUATE PROGRAMME (FYUGP)

BSc PHYSICS HONOURS/MINOR

COMPUTATIONAL PHYSICS

Practical

1. Solution of equations by bisection and Newton-Raphson methods

- Implement the bisection method in Python from scratch.
- Provide at least 4 functions with a specific mathematical equation and find the root using their implementation.
- Analyze and explain the conditions under which the bisection method converges and discuss any potential pitfalls.
- Similarly, implement the Newton-Raphson method in Python.
- Provide the same or different functions and find the root using their implementation.
- Compare the convergence speed of the Newton-Raphson method with the bisection method for different functions.

2. **Least square fitting – straight line fitting**

- Write a code that fits a straight line to the data given and calculates the slope and intercept.
- Plots the regression line along with the data points by giving, labels, title, legends and different colors
- A real-world scenario or dataset can be used to apply linear regression to solve a practical problem.

3. **Numerical Integration – Trapezoidal and Simpson's 1/3 rd rule**

- Implement the Trapezoidal and Simpson's 1/3 Rule in Python for a function given.
- A physics scenario can be provided, where quantities like displacement, work, or energy are needed to calculate through integration. Use both methods to perform the integration and interpret the results.
- Visualize the integration process by plotting the function and the areas under the curve corresponding to the Trapezoidal and Simpson's 1/3 Rule.

4. Simulation of projectile using Euler Method

- Implement projectile motion simulation using the Euler method in Python.
- Simulate the trajectory/ Plot using matplotlib (y vs x, y vs t and x vs t)
- Compare with the theoretical values of range, maximum height and time of flight.

- Change initial conditions such that the projectile is now a freely falling body. Plot y vs t.
- Extend the simulation to include air resistance and compare the projectile motion with and without air resistance.

5. Simulation of simple and damped pendulums using RK2 Method

- Simulates the damped pendulum and stores phase space coordinates to arrays using second order Runge-Kutta method.
- Provide initial conditions and damping parameters for the damped pendulum scenario.
- Plot the motion of the pendulum and phase space trajectories.

  Change the Initial conditions and damping factor and analyse the results. Make sure turning the damping off reproduces the simple pendulum result.

6. Numerical differentiation using difference table.

- Implement numerical differentiation using a difference table in Python.
- Provide a function y = f(x) and a set of data points. Compute the numerical derivative at specific points using the forward difference method.
- Discuss the sensitivity of numerical differentiation to the choice of step size.

  Present physics problems like compute the velocity or acceleration of a particle based on position data.

7. Monte- Carlo simulation of radioactive decay

- Implement a simulation of radioactive decay in Python.
- Provide initial conditions (number of particles, decay constant) and analyze the results, including plotting the decay curve over time.
- Calculate the half-life of the radioactive substance based on the simulation results and check how it compares to the theoretically expected half-life.

  Provide information about a specific radioactive isotope with a known half-life to simulate the decay of this isotope and compare the simulation results with the expected decay.