PH3120 – Computational Physics Laboratory 1

CPL103 – Ordinary Differential Equations

Section 1 & 2

1. **Ordinary Differential Equations (ODE)**

I. What is an ordinary differential equation (ODE)?

II. How is an ODE different from a partial differential equation (PDE)?

III. What are the key components of an ODE?

IV. What is the **order** of an ODE? How does it relate to the number of derivatives present in the equation?

V. What is the general form of a first-order ODE? Give an example.

VI. What is the general form of a second-order ODE? Give an example.

VII. What is an initial value problem (IVP) in the context of ODEs?

2. Euler's Method for Solving ODE

I. Answer the following questions based on the concepts of Euler's method.

(a) What is Euler's method for numerically solving ODEs?

(b) Write the Taylor series expansion for $y(t + dt)$ at t.

(c) Derive an equation to solve an ODE using Euler's methods based on the above series expansion?

(d) What is the local truncation error of the Euler's method?

(e) What is the global truncation error of the Euler's method?

(f) Can you solve ODEs with 2nd or 3rd order using the Euler's method? Explain your answer.

(g) What are the advantages of the Euler's method?

(h) What are the limitations of the Euler's method?

II. Develop a python function with the following signature.

```python
def euler_method(f, x0, t0, dt, num_steps):
    """
    Approximates the solution of an ODE using Euler's method.

    Parameters:
    - f: The derivative function f(t, x) defining the ODE dy/dt = f(t, x).
    - x0: The initial value of x at t0.
    - t0: The initial time value.
    - dt: The time step size.
    - num_steps: The number of steps to take.

    Returns:
    - t: A list of time values.
    - x: A list of approximate solution values.
    """
```

III. The population of bacteria in a container is given by the following expression

$$\dot{x} = ax - bx^2,$$

where $\dot{x} = \frac{dx}{dt}$ and ax is the birth rate and bx is the death rate.

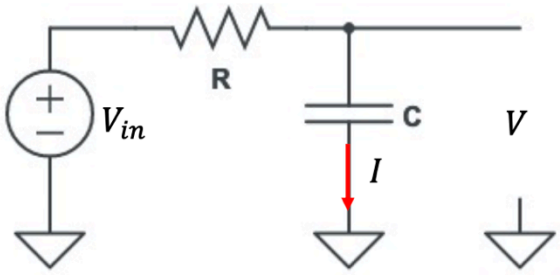
Initially there were 2000 bacteria in the container. Find the number of bacteria after 4 hours if $a = 0.1$ and $b = 0.001$.

(a) Solve the above problem analytically.

(b) Use euler_method() function you developed to solve the problem.

(c) Calculate the percentage error. Comment on the results.

IV. Consider the following circuit diagram which can be used to charge a capacitor.



For the current (I) through capacitor is given by

$$I = C\frac{dV}{dt},$$

and also

$$I = \frac{V_{in} - V}{R}$$

It is given that $V = 0$, when $t = 0$. Furthermore, assume that $V_{in} = 5\text{V}$, $RC = 2\Omega\text{F}$.

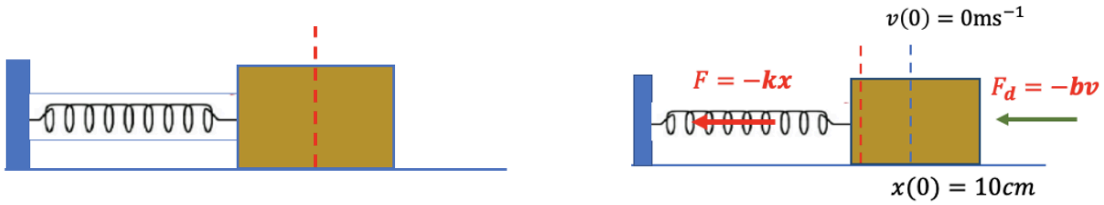
(a) Build an ODE based on the above information.

(b) Solve the differential equation analytically and find V when $t = 4\text{s}$.

(c) Solve the differential equation using euler_method() function and find V when $t = 4\text{s}$.

(d) Calculate the percentage error. Comment on the results.

V. The damped harmonic oscillator refers to a physical system where a mass-spring system experiences damping forces, leading to a gradual reduction in the amplitude of its oscillations over time. The following figure shows a damped oscillator, which was stretched until $x = 10$ cm at $t = 0$ and released. Initially, the oscillator was at rest.



Here, the damping force (F_d) resists the motion of the object. The magnitude of the damping force is directly proportional to the object's velocity ($F_d = -bv$, b is a constant). The direction of the damping force is opposite to the object's velocity.

(a) Obtain the ODE which provides the relationship between acceleration, velocity, and position.

(b) Analytically solve the ODE when $b^2 = 4mk$ and plot $x(t)$ graph.

(c) Modify the Python function in 2 (II) to solve the ODE of a damped harmonic oscillator. Solve the ODE for the three cases in 2 V (b) and estimate the accuracy. Comment on the results.

3. Second Order Runge-Kutta Method for Solving ODE

I. Answer the following questions based on the concepts of the second order Runge-Kutta method.

(a) Derive the formula for calculating the intermediate step in the second order Runge-Kutta method by starting from the Taylor's equation?

(b) What is the truncation error of the second order Runge-Kutta method?

(c) What is the advantage of the second order Runge-Kutta method over the Euler's method?

(d) Explain the concept of Local truncation error and global truncation error in the second order Runge-Kutta method?

(e) How can the step size affect accuracy?

II. Develop a python function with the following signature.

```
def runge_kutta_second_order(f, x0, t0, dt, num_steps):
    """
    Solves a first-order ordinary differential equation using
    the second-order Runge-Kutta method.

    Arguments:
    - f: The derivative function f(t, x) that defines the first-order ODE.
    - x0: The initial value of x at the starting time t0.
    - t0: The initial time value.
    - dt: The time step size.
    - num_steps: The total number of steps to be taken.

    Returns:
    - t: A numpy array containing the time values.
    - x: A numpy array containing the approximate solution values for x.

    """
```

III. Solve the questions in 2 III, IV, V and VI using the second order Runge-Kutta method.

4. Fourth Order Runge-Kutta Method for Solving ODE

I. Answer the following questions based on the concepts of the fourth order Runge-Kutta method.

(a) What is the formula for calculating the intermediate step in the fourth order Runge-Kutta method by starting from the Taylor's equation?

(b) What is the truncation error of the fourth order Runge-Kutta method?

(c) What is the advantage of the fourth order Runge-Kutta method over the Euler's method?

(d) What is the advantage of the fourth order Runge-Kutta method over the second order method?

IV. Develop a python function with the following signature.

```
def runge_kutta_fourth_order(f, x0, t0, dt, num_steps):
    """
    Solves a first-order ordinary differential equation using
    the fourth-order Runge-Kutta method.

    Arguments:
    - f: The derivative function f(t, x) that defines the first-order ODE.
    - x0: The initial value of x at the starting time t0.
    - t0: The initial time value.
    - dt: The time step size.
    - num_steps: The total number of steps to be taken.

    Returns:
    - t: A numpy array containing the time values.
    - x: A numpy array containing the approximate solution values for x.

    """
```

V. Solve the questions in 2 III, IV, and V using the fourth order Runge-Kutta method.

5. Python ODE Solvers

I. Solve the initial values problems in 2 III, IV, and V using the ODE solver (solve_ivp() function) in the Scipy package of Python.

II. Provide a comprehensive analysis of the accuracy of the solutions obtained from Euler's method, the second-order and fourth-order Runge-Kutta methods, and the solve_ivp() function. Conclude by determining which method is more accurate and efficient for the given problems in 2 III, IV, and V.