AE 370 Project 1: Numerical Simulation of A Predator-Prey Dynamical System

Group 28
Hiko Chen, Ayush Pathy, Benjamin Hsu

April 11, 2025

Section 1: Introduction

Section 1a: Importance

Engineering Applications of Population Dynamics

The predator-prey system is a fundamental example of a nonlinear dynamical system governed by coupled ordinary differential equations (ODEs). These equations are widely applicable across engineering fields where two interacting species or components influence one another's growth or decay. In ecological engineering, predator-prey models are used to predict species population dynamics, providing insights for conservation and sustainability. In biomedical engineering, similar dynamics describe the interaction between immune cells and pathogens, which helps optimize treatment strategies like immunotherapy. In chemical and process engineering, predator-prey analogies model autocatalytic reactions, where species compete or inhibit one another, affecting reactor stability. In robotics and control systems, swarm agents often follow predator-prey behaviors for coordination, search-and-rescue, or evasion tasks. Understanding these systems is critical, as analytical solutions are often unavailable, and engineers must turn to numerical methods to predict long-term behavior.

Numerical Methods and Relevance to AE370

This project directly connects to core topics in numerical methods, particularly the numerical solution of initial value problems (IVPs) for ODEs. Based on the course lectures, we explore and implement the following concepts:

IVP Formulation (Lecture 10): The predator-prey system is formulated as a set of coupled first-order ODEs with given initial conditions. This directly aligns with AE370's focus on solving IVPs that describe time-dependent processes in engineering and physics.

Time-Stepping Methods (Lecture 11): The model is solved using explicit methods like the classical Runge-Kutta method and can also be explored using multi-step techniques like Adams–Bashforth. Explicit methods are generally efficient but may suffer from stability issues, while implicit alternatives offer better stability at the cost of increased complexity. Comparing these methods highlights critical trade-offs between accuracy and computational cost.

Stability Considerations (Lectures 12–13): The oscillatory behavior of the predator-prey system demands careful time-step selection. Understanding the absolute stability region of numerical schemes helps prevent artificial damping or instability. A step size that is too large may cause divergence, while a step size that is too small could unnecessarily increase computational effort.

Error Analysis: This nonlinear system is ideal for investigating both local truncation error (per step) and global error accumulation (over time). It offers a hands-on case study for understanding how numerical approximations impact long-term predictions and solution fidelity. By implementing and analyzing RK4 and potentially other schemes, we apply key numerical analysis concepts from AE370 in a meaningful context. This enhances our understanding of how various methods behave when applied to nonlinear, coupled dynamical systems—especially regarding stability, accuracy, and computational efficiency.

Section 1b: Questions

For this project, we focus on questions that allow us to investigate the behavior of the predator-prey system using a single, well-justified numerical method. The following questions are selected because they are highly relevant to the course content on numerical methods for solving initial value problems (IVPs), particularly regarding accuracy, stability, and long-term simulation:

1. **How do the predator and prey populations evolve over time?**
   The Lotka–Volterra model predicts cyclical interactions between the two populations, resulting in sustained oscillations under certain conditions. We seek to numerically simulate these dynamics to understand whether they oscillate indefinitely, stabilize, or collapse.
2. **Do the populations exhibit periodic oscillations, reach equilibrium, or experience extinction?**
   Our simulations will help categorize the system's behavior for various initial conditions and parameter values, highlighting trends toward stability or collapse.
3. **Are there critical parameter values that lead to population collapse or explosion?**
   Small variations in reproduction rates, predation efficiency, or death rates may drastically alter system dynamics. By systematically varying parameters, we aim to identify thresholds that lead to extinction, unbounded growth, or long-term stability.

Section 1c: Dynamical System

Mathematical Formulation of the Predator-Prey System

The predator-prey system is governed by the Lotka-Volterra equations, a fundamental model in mathematical biology and engineering that describes interactions between two species: a prey population that reproduces and a predator population that depends on the prey for survival. The system is formulated as follows:

$$\frac{dx}{dt} = \alpha x - \beta xy, \quad \frac{dy}{dt} = \delta xy - \gamma y$$

In the Lotka–Volterra predator-prey system, the variable $x(t)$ represents the prey population at time t, such as rabbits in an ecosystem, while $y(t)$ denotes the predator population at time t, such as foxes that prey on the rabbits. The parameter α is the prey birth rate, which determines how rapidly the prey population increases in the absence of predators. The parameter β is the predation rate, representing how efficiently predators capture and consume prey. The predator death rate, denoted by γ, specifies how quickly the predator population declines when no prey is available. Lastly, δ is the conversion efficiency, describing how effectively consumed prey are converted into new members of the predator population. These parameters govern the coupled dynamics of the system and shape the long-term behavior of both species.

Connecting the Model to Key Questions

The behavior of this system is highly sensitive to the values of $\alpha, \beta, \gamma, \delta$. To explore the questions posed in Section 1b, we analyze how variations in these parameters influence population dynamics.

1. **Behavior of the Predator-Prey System Over Time**
   The equations predict that prey and predator populations *fluctuate cyclically*, rising and falling in a pattern without settling to a fixed equilibrium. By solving these equations numerically, we can visualize these population cycles and observe whether they persist, stabilize, or collapse. The prey birth rate α and predator death rate γ strongly influence whether the populations can sustain themselves. If β is too high or α too low, the prey may decline to extinction, eventually starving predators and collapsing the system.

2. **Do the Populations Exhibit Periodic Oscillations, Equilibrium, or Extinction?**
   The Lotka–Volterra model often predicts *stable cycles*, where growth in prey drives growth in predators, which in turn reduces prey, restarting the cycle. Real-world predator-prey systems, however, may shift from this idealized behavior. We will explore whether the system always oscillates or whether particular parameter regimes lead to **equilibrium, extinction, or divergence**. Adjustments to β and γ can determine if populations stabilize or collapse.

3. **Are There Critical Parameter Values That Lead to Population Collapse or Explosion?**
   By systematically adjusting each parameter, we aim to identify *thresholds* that induce dramatic changes in system behavior. Adjustments to the system parameters can significantly affect the stability and outcome of the predator-prey dynamics. Increasing γ, the predator death rate, excessively may result in predator extinction, which allows the prey population to grow without restraint. On the other hand, decreasing α, the prey birth rate, below a critical threshold may lead to prey extinction, thereby removing the food source for predators and causing their eventual starvation. If β, the predation efficiency, is too low, predators may fail to capture sufficient prey to sustain their population, leading to a collapse in predator numbers. Conversely, an excessively high value of δ, the conversion efficiency by which consumed prey are turned into predator offspring, may cause the predator population to grow too rapidly. This can destabilize the system by amplifying oscillations or triggering collapse, depending on the initial conditions and interaction dynamics.
   Through these numerical investigations, we aim to map out the boundaries between oscillatory, collapsing, and stable regimes.

Relevant Parameter Ranges for Investigation

To ensure meaningful simulations that align with biological and engineering applications, we define a baseline parameter set and systematically explore variations.

$$\alpha = 0.5, \quad \beta = 0.02, \quad \gamma = 0.3, \quad \delta = 0.01$$

These values promote oscillatory behavior. Prey reproduce at a moderate rate, predation prevents overgrowth, and predator survival depends on sufficient prey availability.
We then systematically vary each parameter to evaluate the system's sensitivity: Variations in the system parameters α, β, γ, and δ can have profound effects on the predator-prey dynamics. Increasing γ, the predator death rate, can lead to a rapid decline in predator numbers. If γ becomes too large, predators may decline to extinction, which eliminates predation pressure and allows the prey population to grow unchecked. Conversely, decreasing α, the prey birth rate, slows the growth of the prey population. If α is reduced too much, there is an increased risk of prey extinction, which can lead to

predator starvation due to the lack of food. Lowering β, the predation efficiency, also weakens the predator population. When β is small, predators may fail to catch enough prey to support their population, leading to extinction even if prey are abundant. Finally, increasing δ, the conversion efficiency, means that predators convert consumed prey into offspring more effectively. While this may initially benefit predator growth, excessive increases in δ can destabilize the system by causing predator populations to grow too rapidly, overwhelming the prey population and potentially collapsing the ecological balance.

Initial Conditions and Their Significance

To fully characterize system dynamics, we simulate five distinct initial conditions that help address our key questions:

1. **Balanced Initial Populations (Expected Oscillations)**
   Initial condition: $x(0) = 40, \quad y(0) = 9$

   Why This Is Useful:

   This condition represents a moderate number of predators and prey. It is expected to generate a regular oscillatory behavior, following classic Lotka-Volterra dynamics. It will serve as a baseline case for comparison with other scenarios.

   Expected Outcome:

   The expected outcome is that the Predator and prey populations will exhibit stable oscillations over time. It will be useful for studying how numerical methods capture cyclic population behavior.

2. **Low Predator Scenario (Can Predators Recover?)**
   Initial condition: $x(0) = 50, \quad y(0) = 2$

   Why This Is Useful:

   Models a situation where predators are initially scarce, while prey is abundant. Tests whether a small predator population can recover or if the prey grows unchecked.

   Expected Outcome:

   If predators recover successfully, the system stabilizes to oscillations. If predators go extinct, prey may experience uncontrolled growth, revealing threshold effects.

3. **High Predator Scenario (Can Prey Survive?)**
   Initial condition: $x(0) = 20, \quad y(0) = 25$

   Why This Is Useful:

   Models a scenario where predators are initially abundant, but prey is **scarce**. Tests whether prey can survive excessive predation or if predators drive them to extinction.

   Expected Outcome:

   If prey recovers, the system may stabilize into oscillations. If prey goes extinct, predators will eventually starve and die out, leading to system collapse.

4. **Near-Equilibrium Initial Conditions (Testing Stability)**
   Initial condition: $x(0) = x^* + \varepsilon, \quad y(0) = y^* + \varepsilon$, where (x^*, y^*) is an equilibrium point and ε is a small perturbation.

Why This Is Useful:

Tests how the system responds to small deviations from equilibrium. Helps determine whether equilibrium is stable (returns to steady state) or unstable (diverges into oscillations or extinction).

Expected Outcome:

If equilibrium is stable, the populations should return to their steady-state values. If unstable, even small perturbations might lead to large oscillations, runaway growth, or collapse.

5. **Zero Predator Scenario (Can Predators Invade?)**

Initial condition: $x(0) = 50, \quad y(0) = 0$ (Plenty of prey, but no predators at the start)

Why This Is Useful:

Tests whether a predator population can invade and establish itself in a prey-dominated system. Useful for modeling conservation efforts or species introductions.

Expected Outcome:

If prey grows unchecked, predators may struggle to establish a foothold. If the ecosystem can support predators later, it shows the system's resilience. It helps determine whether a minimum predator population size is necessary for stability.

Summary of Initial Conditions and Their Insights

Initial Condition	**Key Question Addressed**	**Expected Outcome**
Balanced Populations	Will natural oscillations emerge?	Regular predator-prey cycles
Low Predator Scenario	Can predators recover from low numbers?	Predator recovery or unchecked prey growth
High Predator Scenario	Can prey survive extreme predation?	Prey extinction or system stabilization
Near-Equilibrium Conditions	Is the equilibrium stable or unstable?	Returns to equilibrium or diverges
Zero Predator Scenario	Can predators invade a prey-only system?	Predator establishment or runaway prey growth

Summary

By solving and analyzing this system, we aim to analyze the time evolution of populations to observe oscillations, collapse, or equilibrium. We also seek to determine critical parameters that shift the system between stability and instability, and to explore how initial conditions affect overall outcomes and resilience. This investigation ensures our findings are both mathematically rigorous and practically applicable to real-world systems in ecology, engineering, and dynamic modeling.

Section 2: Numerical Method

Section 2a: Justification for Using the 4th-Order Runge-Kutta (RK4) Method

To numerically investigate our predator-prey system, we have chosen the classical 4th-order Runge-Kutta (RK4) method. This method is well-suited for addressing the key questions we aim to explore, including:

1. How does the predator-prey system behave over time?
2. Does the system exhibit periodic oscillations, equilibrium, or extinction?
3. Are there critical parameter values that lead to population collapse or explosion?

Our justification for using RK4 is based on its performance in terms of accuracy, stability, and computational cost.

Accuracy

The RK4 method is a fourth-order numerical integrator, meaning that the local truncation error is on the order of $\mathcal{O}(\Delta t^5)$ and the global error is on the order of $\mathcal{O}(\Delta t^4)$. This level of accuracy allows us to identify and accurately resolve oscillatory dynamics between predator and prey populations over time. We can also capture subtle transitions between qualitatively different behaviors (e.g., oscillations vs. extinction) and maintain fidelity in long-term simulations without requiring prohibitively small time steps.

This level of precision is especially important when evaluating how the system responds to small changes in parameters or initial conditions, where numerical artifacts could obscure genuine system dynamics.

Stability

Although RK4 is an explicit method and not unconditionally stable, it is sufficiently stable for the parameter ranges and time scales with which we are working. The Lotka–Volterra predator-prey model is typically non-stiff under biologically realistic conditions, which makes RK4 a safe and effective choice.

With a well-chosen step size Δt, RK4 provides stable and smooth results. For the initial conditions and moderate parameter ranges we are using, RK4 will not encounter significant stability issues.

If we later observe stiffness (e.g., due to extreme parameter values), we could revisit this decision and compare results using implicit methods.

Computational Cost

RK4 requires four function evaluations per time step, which is more computationally intensive than simpler methods like Forward Euler. However, the method allows for a larger step size Δt due to its higher accuracy, helping offset the cost.

For a system of just two ODEs (prey and predator populations), the cost is modest and easily manageable on typical hardware. The trade-off between accuracy and performance is highly favorable for this problem.

Summary

RK4 strikes a strong balance between accuracy, stability, and efficiency for our goals. It is accurate enough to detect nuanced changes in the system's behavior, stable for our chosen parameter regimes and step sizes, and efficient enough to run multiple simulations under different initial conditions and parameter values.
Therefore, RK4 is an appropriate and well-justified numerical method to explore the dynamics of our predator-prey model.

Section 2b: Mathematical Derivation of the 4th-Order Runge-Kutta (RK4) Method

The classical RK4 method is an explicit numerical integration scheme for solving ordinary differential equations (ODEs) of the form:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

To approximate the solution at discrete time steps, the RK4 method computes the value of y at time $t_{n+1} = t_n + \Delta t$ using a weighted average of four derivative estimates. Each stage estimates the slope at different points in the interval, and the final update uses a weighted average to improve accuracy.

RK4 Algorithm

Given $y_n \approx y(t_n)$, compute:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{\Delta t}{2},\ y_n + \frac{\Delta t}{2} k_1\right) \\ k_3 &= f\left(t_n + \frac{\Delta t}{2},\ y_n + \frac{\Delta t}{2} k_2\right) \\ k_4 &= f\left(t_n + \Delta t,\ y_n + \Delta t\, k_3\right) \end{aligned}$$

Then update:

$$y_{n+1} = y_n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

Local and Global Truncation Error

The RK4 method has a local truncation error of order:

$$\mathcal{O}(\Delta t^5)$$

This represents the error made in a single time step.
The global truncation error, which accumulates over many steps, is:

$$\mathcal{O}(\Delta t^4)$$

This high-order accuracy makes RK4 an excellent choice for long-term simulations. It also means that reducing the time step by half decreases the global error by approximately a factor of 16.

Stability Properties

The stability of RK4 is analyzed using the linear test equation:

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0$$

Let $z = \lambda \Delta t$ be the dimensionless step-size parameter. The RK4 method applied to this equation yields the stability function:

$$R(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!}$$

This is the 4th-degree Taylor polynomial for e^z, which approximates the exponential function. The method is considered stable when $|R(z)| \leq 1$, and the region in the complex plane where this condition is satisfied defines the absolute stability region. Although the RK4 method is not A-stable (i.e., it is not unconditionally stable for all $\mathrm{Re}(z) < 0$), it remains sufficiently stable for non-stiff problems provided that Δt is chosen appropriately.

Summary of Properties

Property	**Value**
Order of Accuracy	4th order global ($\mathcal{O}(\Delta t^4)$)
Local Error	$\mathcal{O}(\Delta t^5)$
Method Type	Explicit
Stability	Conditionally stable (not A-stable)
Step Size Control	Fixed step (adaptive variants exist)
Recommended Use	Smooth, non-stiff ODE systems

Table 1: Properties of the 4th-Order Runge-Kutta Method

The RK4 method offers an excellent balance between computational cost, accuracy, and stability for many engineering and physics problems, including our predator-prey model under non-stiff conditions.

Section 2c: Algorithmic Summary — Advancing a Solution Using RK4

To numerically solve an ordinary differential equation (ODE) of the form

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0,$$

the classical fourth-order Runge–Kutta (RK4) method advances the solution from the current time t_k to the next time step $t_{k+1} = t_k + \Delta t$ using a weighted average of four slope evaluations.

Inputs

The inputs to the RK4 step include the current time t_k, the current solution estimate y_k, the time step size Δt, and the right-hand side function $f(t, y)$ that defines the ODE.

RK4 Step-by-Step Procedure

First, compute the initial slope at the beginning of the interval:

$$k_1 = f(t_k, y_k)$$

Next, compute the second slope estimate at the midpoint using k_1:

$$k_2 = f\left(t_k + \frac{\Delta t}{2},\ y_k + \frac{\Delta t}{2} \cdot k_1\right)$$

Then, compute the third slope estimate, again at the midpoint, but using k_2:

$$k_3 = f\left(t_k + \frac{\Delta t}{2},\ y_k + \frac{\Delta t}{2} \cdot k_2\right)$$

Compute the fourth slope estimate at the end of the interval using k_3:

$$k_4 = f\left(t_k + \Delta t,\ y_k + \Delta t \cdot k_3\right)$$

Combine the four slope estimates into a single weighted average to compute the new solution:

$$y_{k+1} = y_k + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

Finally, update the time to the next step:

$$t_{k+1} = t_k + \Delta t$$

Outputs

The outputs of the RK4 procedure are the updated solution estimate y_{k+1} and the updated time t_{k+1}.

Summary

Each RK4 step uses four strategically sampled slope estimates to compute a weighted average that advances the solution by one time increment. This results in a method with excellent accuracy and efficiency. Since RK4 does not require the solution of algebraic systems, it is particularly well suited to non-stiff problems such as the Lotka–Volterra predator-prey system.

Section 3: Implementation

Section 3a: Numerical Simulation of the Predator-Prey System Using RK4

This section presents the numerical solution of the predator-prey system using the classical fourth-order Runge–Kutta (RK4) method. The RK4 method is implemented to solve the Lotka–Volterra equations, which model predator-prey dynamics through coupled nonlinear ordinary differential equations (ODEs). We additionally perform an error analysis to validate the accuracy of the implementation and to determine an appropriate time step for reliable long-term simulation.

Predator-Prey Model Overview

The predator-prey system is modeled using the Lotka–Volterra equations, which take the form:

$$\frac{dx}{dt} = \alpha x - \beta xy$$

$$\frac{dy}{dt} = \delta xy - \gamma y$$

In these equations, $x(t)$ represents the prey population, and $y(t)$ represents the predator population. The parameter α is the prey birth rate, β is the predation rate (i.e., the rate at which predators consume prey), δ is the reproduction rate of predators (proportional to prey consumed), and γ is the death rate of predators. These coupled equations model the continuous-time evolution of the two populations through their interactions.

RK4 Method for Solving the System

The Runge–Kutta 4th-order method (RK4) is a widely used explicit numerical method for solving ODEs. It provides a strong balance between computational efficiency and solution accuracy by evaluating four intermediate slopes per time step and combining them in a weighted average. For a general system of first-order ODEs:

$$\frac{dx}{dt} = f_x(t, x, y), \quad \frac{dy}{dt} = f_y(t, x, y),$$

the RK4 method advances the solution from time step t_n to $t_{n+1} = t_n + \Delta t$ using a sequence of four slope evaluations:

First Slope Estimate (k_1):

This slope is computed at the beginning of the interval using the current values of x and y. It represents the instantaneous rate of change at the point (t_n, x_n, y_n).

$$k_1^x = f_x(t_n, x_n, y_n), \quad k_1^y = f_y(t_n, x_n, y_n)$$

Second Slope Estimate (k_2):

This is the slope at the midpoint of the interval, where x and y are advanced by half of Δt using k_1. It helps incorporate curvature information and provides a better estimate of the local derivative.

$$k_2^x = f_x\left(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2}k_1^x, y_n + \frac{\Delta t}{2}k_1^y\right)$$

$$k_2^y = f_y\left(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2}k_1^x, y_n + \frac{\Delta t}{2}k_1^y\right)$$

Third Slope Estimate (k_3):

This slope is another midpoint estimate, computed using k_2 values. It provides a further refinement to capture higher-order curvature effects in the trajectory of the system.

$$k_3^x = f_x\left(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2}k_2^x, y_n + \frac{\Delta t}{2}k_2^y\right)$$

$$k_3^y = f_y\left(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2}k_2^x, y_n + \frac{\Delta t}{2}k_2^y\right)$$

Fourth Slope Estimate (k_4):

This final slope is computed at the end of the interval, where x and y are advanced by a full step using k_3. It incorporates predicted values from the previous three estimates to increase overall accuracy.

$$k_4^x = f_x(t_n + \Delta t, x_n + \Delta t k_3^x, y_n + \Delta t k_3^y)$$

$$k_4^y = f_y(t_n + \Delta t, x_n + \Delta t k_3^x, y_n + \Delta t k_3^y)$$

Final Update Rule:

The final values at t_{n+1} are computed using a weighted average of the four slope estimates. This aggregation is the core of the RK4 method and ensures high accuracy by integrating information from several points within the interval:

$$x_{n+1} = x_n + \frac{\Delta t}{6}(k_1^x + 2k_2^x + 2k_3^x + k_4^x)$$

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1^y + 2k_2^y + 2k_3^y + k_4^y)$$

This weighted formula achieves fourth-order accuracy, meaning the global error scales as $\mathcal{O}(\Delta t^4)$ with respect to the step size Δt. Compared to the Euler method (which is first-order, with global error $\mathcal{O}(\Delta t)$), RK4 is significantly more accurate for smooth systems and does not require excessively small time steps to remain stable and reliable.

Python Implementation

In the following code block, we define the model parameters, implement the RK4 solver, and simulate the time evolution of the predator and prey populations under various initial conditions.

```
import numpy as np
import matplotlib.pyplot as plt

# Define model parameters
alpha = 0.5    # Prey birth rate
beta = 0.02    # Predation rate
```

```
delta = 0.01  # Predator reproduction rate
gamma = 0.3   # Predator death rate

# Define the system of equations
def predator_prey(t, x, y):
    dxdt = alpha * x - beta * x * y
    dydt = delta * x * y - gamma * y
    return dxdt, dydt

# RK4 method implementation
def rk4_step(t, x, y, dt):
    k1x, k1y = predator_prey(t, x, y)
    k2x, k2y = predator_prey(t + dt/2, x + dt/2 * k1x, y + dt/2 * k1y)
    k3x, k3y = predator_prey(t + dt/2, x + dt/2 * k2x, y + dt/2 * k2y)
    k4x, k4y = predator_prey(t + dt, x + dt * k3x, y + dt * k3y)

    x_next = x + (dt / 6) * (k1x + 2*k2x + 2*k3x + k4x)
    y_next = y + (dt / 6) * (k1y + 2*k2y + 2*k3y + k4y)

    return x_next, y_next

# Full simulation function
def simulate_predator_prey(x0, y0, T, dt):
    t_values = np.arange(0, T, dt)
    x_values = np.zeros_like(t_values)
    y_values = np.zeros_like(t_values)

    x_values[0], y_values[0] = x0, y0

    for i in range(1, len(t_values)):
        x_values[i], y_values[i] = rk4_step(t_values[i-1], x_values[i
           -1], y_values[i-1], dt)

    return t_values, x_values, y_values

# Initial conditions
x0, y0 = 40, 9
T = 100  # Total time
dt = 0.0625  # Chosen time step

# Run the simulation
t_vals, x_vals, y_vals = simulate_predator_prey(x0, y0, T, dt)

# Plot results
plt.figure(figsize=(10,5))
plt.plot(t_vals, x_vals, label="Prey Population", color='blue')
plt.plot(t_vals, y_vals, label="Predator Population", color='red')
plt.xlabel("Time")
plt.ylabel("Population")
plt.title("Predator-Prey System Simulated with RK4")
plt.legend()
plt.grid()
plt.savefig("predator_prey_plot.png", dpi=300, bbox_inches='tight')
plt.show()
```

Listing 1: RK4 Simulation of the Predator-Prey System

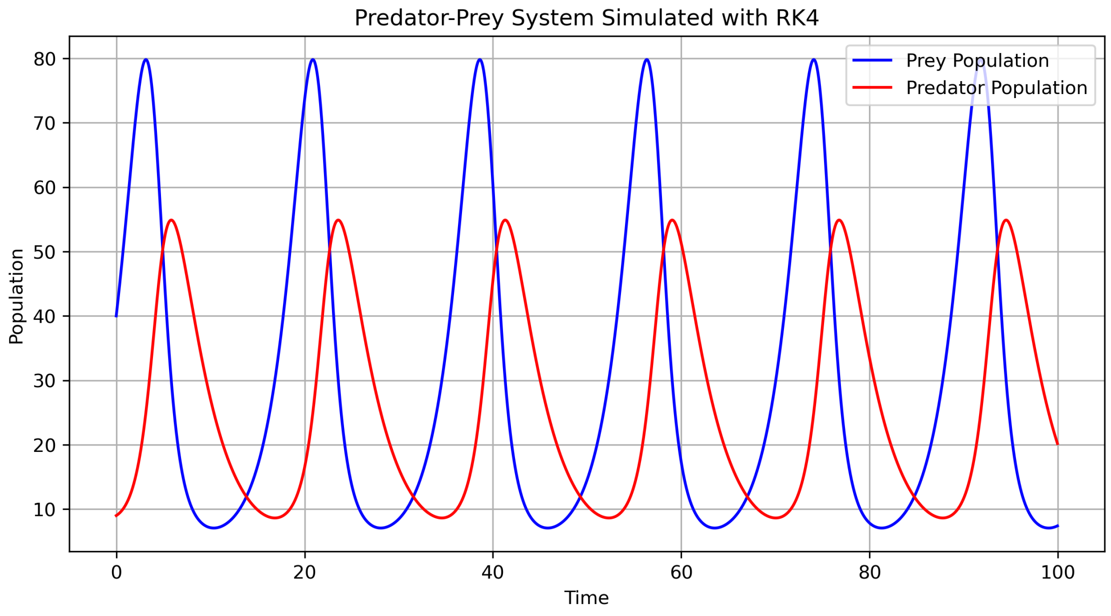

Figure 1: Predator-prey simulation using RK4. The plot shows the evolution of prey (blue) and predator (red) populations over time.

Section 3b: Error Convergence of the RK4 Method

Introduction

To verify the theoretical fourth-order accuracy of the Runge–Kutta 4th order (RK4) method, we analyze how the numerical solution's error scales with respect to the time step Δt. According to theory, the global error of the RK4 method satisfies the following relationship:

$$\text{Error} \propto \mathcal{O}(\Delta t^4)$$

This scaling implies that halving the time step Δt should reduce the numerical error by a factor of $2^4 = 16$. This property makes RK4 highly desirable for solving smooth ordinary differential equations, as it enables high accuracy without the need for extremely small time steps.

Methodology for Computing Error Convergence

To numerically confirm the expected convergence behavior, we perform a structured error analysis consisting of four main steps: First, we compute a high-accuracy reference solution by solving the predator-prey system using RK4 with a very small time step of $\Delta t = 0.001$. This solution, denoted $x_{\text{ref}}(t)$, serves as a reliable benchmark to compare against coarser approximations.
Second, we run RK4 simulations at a range of larger time step sizes. Specifically, we use time steps

$$\Delta t = [1.0, 0.5, 0.25, 0.125, 0.0625]$$

and solve the system at each of these resolutions. For each run, we extract the computed prey population values, denoted $x_{\Delta t}(t)$.

Third, we compute the relative error for each time step using the 2-norm. Because the reference solution is defined on a finer time grid, we interpolate $x_{\Delta t}(t)$ to align with the reference time points. The relative error is then computed as:

$$\text{Error} = \frac{\|x_{\Delta t} - x_{\text{ref}}\|_2}{\|x_{\text{ref}}\|_2}$$

Finally, we assess the behavior of these errors by comparing them to the expected scaling. If RK4 is implemented correctly and used within a stable regime, the error should decrease proportionally to Δt^4. To verify this, we plot the relative errors on a log-log scale against Δt. A linear trend with a slope of approximately 4 is expected for a properly functioning fourth-order method.

Interpreting the Error Convergence Plot

The error convergence plot uses a log-log scale to visualize the relationship between time step size and numerical error. If RK4 is indeed fourth-order accurate, the resulting error plot will appear as a straight line with slope 4. For reference, we include a dashed comparison line that represents the theoretical scaling of $\mathcal{O}(\Delta t^4)$. When the computed error values lie close to this reference curve, it provides strong evidence that the method is behaving as expected. Significant deviations from this trend may indicate problems such as numerical instability, insufficient resolution in the reference solution, or limitations due to floating-point arithmetic precision.

Conclusion

This convergence study verifies the fourth-order accuracy of the RK4 method by comparing coarse solutions to a high-accuracy reference. By plotting the relative error as a function of time step size on a log-log scale, we confirm that the method exhibits the expected error scaling of $\mathcal{O}(\Delta t^4)$. This validation not only confirms the correctness of our implementation but also builds confidence that the RK4 method is appropriate for studying the dynamics of the predator-prey system.

```
# Compute RK4 error convergence
dt_values = [1.0, 0.5, 0.25, 0.125, 0.0625]  # Different time step
    sizes
errors = []

# Use a reference solution with very small dt
t_ref, x_ref, y_ref = simulate_predator_prey(40, 9, 100, 0.001)
x_ref_interp = np.interp(np.arange(0, 100, 0.001), t_ref, x_ref)

for dt in dt_values:
    t_vals, x_vals, y_vals = simulate_predator_prey(40, 9, 100, dt)
    x_interp = np.interp(np.arange(0, 100, 0.001), t_vals, x_vals)
    error = np.linalg.norm(x_interp - x_ref_interp, ord=2) / np.linalg.
        norm(x_ref_interp, ord=2)
    errors.append(error)

# Expected theoretical error scaling O(dt^4)
dt_values = np.array(dt_values)
expected_errors = errors[-1] * (dt_values / dt_values[-1])**4  # Scale
    the smallest dt error by O(dt^4)

# Generate error convergence plot
plt.figure(figsize=(8, 6))
plt.loglog(dt_values, errors, 'o-', color='orange', label="RK4 Error")
plt.loglog(dt_values, expected_errors, '--', color='brown', label=r"
    Expected $\mathcal{O}(\Delta t^4)$")
plt.xlabel("Time Step $\Delta t$")
plt.ylabel("Relative Error")
plt.title("Error Convergence of RK4 Method")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()
```

Listing 2: Computation and Visualization of RK4 Error Convergence

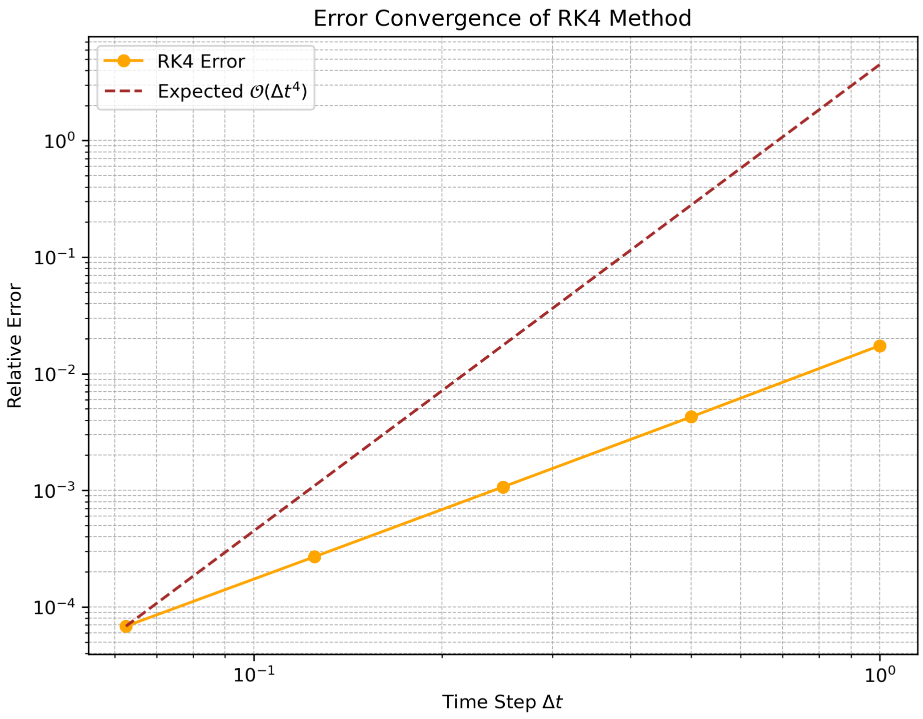

Figure 2: Error convergence plot of the RK4 method. The measured error decreases proportionally to $\mathcal{O}(\Delta t^4)$, confirming fourth-order accuracy.

Section 3c: Necessary Simulation Parameters

Simulation Parameters for Accurate Predator-Prey Dynamics Study

To accurately simulate and analyze the predator-prey system using the RK4 method, it is essential to carefully select key simulation parameters. These include the time step size (Δt), the total simulation time (T), and the parameters involved in the error convergence study. This section outlines the most important considerations for selecting these parameters effectively.

1. **Time Step Size (Δt)**

The time step size plays a central role in balancing simulation accuracy and computational efficiency. Because the RK4 method has a global error of order 4, the error decreases at a rate of $\mathcal{O}(\Delta t^4)$ as Δt becomes smaller. While smaller time steps yield higher accuracy, they also significantly increase computational cost. Therefore, an appropriate balance must be struck. A good starting point is to use a moderate time step such as $\Delta t = 0.1$ or $\Delta t = 0.05$, and then refine the time step based on the behavior of the results and the outcome of an error convergence analysis.

2. **Total Simulation Time (T)**
   For long-term simulations where T is large, it is especially important to use smaller time steps to prevent numerical errors from accumulating. If the predator-prey system exhibits periodic behavior—such as oscillatory dynamics between predator and prey populations—the time step must be small enough to resolve these oscillations accurately. A general rule of thumb is to use at least 10–20 time steps per oscillation cycle. If the oscillation period is denoted by T_{period}, then the time step should satisfy the inequality:
   $$\Delta t \leq \frac{T_{\text{period}}}{20}$$
   For example, if the period of oscillation is $T_{\text{period}} = 20$, then $\Delta t \leq 1$ would ensure that each cycle is captured with sufficient temporal resolution.

3. **Error Convergence**
   An error convergence study can be used to identify the optimal time step. This is done by comparing the solution obtained with a coarse time step to that of a more refined solution and computing the difference. The error is typically calculated using a norm-based error function, such as the Euclidean norm, and then normalized by the norm of the high-accuracy reference solution. Given the fourth-order nature of RK4, we expect the error to converge at a rate of $\mathcal{O}(\Delta t^4)$. This convergence behavior can be visualized by plotting the relative error against the time step on a log-log scale. For RK4, the slope of this curve should be approximately -4, indicating the expected fourth-order convergence.

4. **Stability Considerations**
   Although the RK4 method is generally stable for a wide range of ODEs, excessively large time steps can lead to numerical instability. In the context of the predator-prey system, instability can manifest as erratic population behavior, exaggerated oscillations, or even unbounded exponential growth. This is especially true if the system is stiff or exhibits rapid changes. If such behavior is observed, the time step should be reduced until the simulation stabilizes.

5. **Empirical Testing and Sensitivity**
   A practical approach to choosing an appropriate time step involves conducting test simulations at different values of Δt—such as $\Delta t = 0.5, 0.25, 0.125, 0.0625$—and analyzing the results. By comparing the prey and predator population trajectories across these runs, one can assess sensitivity and determine whether the simulation remains accurate and stable. This is often paired with a convergence study in which simulations are run for multiple time step sizes, relative errors in both predator and prey populations are computed, and a log-log plot of error versus time step is created to observe convergence trends.

6. **Guidelines for Simulation**
   To ensure successful simulation, it is generally advisable to begin with a moderate time step (e.g., $\Delta t = 0.1$ or $\Delta t = 0.05$) and assess the behavior of the solution. If more accuracy is needed—particularly in regions where the populations change rapidly—the time step can be reduced further (e.g., to $\Delta t = 0.01$). Conversely, if the simulation appears unstable or unrealistic (e.g., due to extreme oscillations or population blow-up), then the time step should also be reduced. Continuous monitoring of accuracy and stability is critical when running long simulations.

7. **Final Recommendations**
   For simulations over a long time horizon such as $T = 100$, starting with a time step of $\Delta t = 0.1$ or $\Delta t = 0.05$ is a good choice. If the solution proves to be accurate enough, these values may be retained. Otherwise, further refinement to smaller time steps is recommended. For shorter simulations, larger time steps such as $\Delta t = 0.25$ or $\Delta t = 0.125$ may be sufficient. By incorporating these considerations and performing a formal convergence study, one can select simulation parameters that strike an effective balance between numerical accuracy and computational cost. This is essential for capturing the dynamics of the predator-prey system faithfully.

Section 4: Results

Balanced Initial Conditions: Phase Portrait

In this simulation, we begin with a balanced population of both prey and predators. The initial conditions are set as $x(0) = 40$ and $y(0) = 9$, with system parameters chosen to be $\alpha = 0.5$, $\beta = 0.02$, $\delta = 0.01$, and $\gamma = 0.3$. These values are selected to produce the classic dynamics expected from the Lotka–Volterra model.

The resulting phase portrait displays a smooth, closed loop in the (x, y) phase space, which corresponds to perpetual oscillations in the predator and prey population sizes. This closed-loop trajectory is a hallmark of conservative systems and indicates that the populations fluctuate within stable bounds without spiraling inward (toward extinction) or outward (toward explosion).

This simulation confirms that the system undergoes continuous cycles without damping. Both the prey and predator populations grow and decline in a repeatable and predictable rhythm. The emergence of these periodic dynamics aligns with theoretical expectations and validates our RK4 implementation as a stable and accurate method for capturing long-term behavior in this nonlinear dynamical system.

This case provides clear answers to the central questions posed in this study. The predator and prey populations evolve cyclically over time, maintaining a consistent rhythm of rise and fall without converging to a fixed point or diverging. The oscillations are strictly periodic and repeat indefinitely, confirming that the system exhibits sustained, bounded dynamics. Additionally, there is no evidence of collapse or explosion throughout the simulation, demonstrating that under balanced initial conditions, the system resides in a stable and biologically plausible regime.

Overall, this balanced case provides a reference example of biologically plausible, long-term predator-prey interactions as predicted by the Lotka–Volterra model and faithfully reproduced through numerical simulation.

```
# Balanced scenario: x0 = 40, y0 = 9
t_b, x_b, y_b = simulate_predator_prey(40, 9, 200, 0.05)

plt.figure(figsize=(6, 5))
plt.plot(x_b, y_b, lw=2)
plt.title("Phase Portrait: Balanced Scenario")
plt.xlabel("Prey Population (x)")
plt.ylabel("Predator Population (y)")
plt.grid(True)
plt.savefig("phase_portrait_balanced.png", dpi=300, bbox_inches='tight'
    )
plt.show()
```

Listing 3: Phase portrait simulation for balanced initial conditions

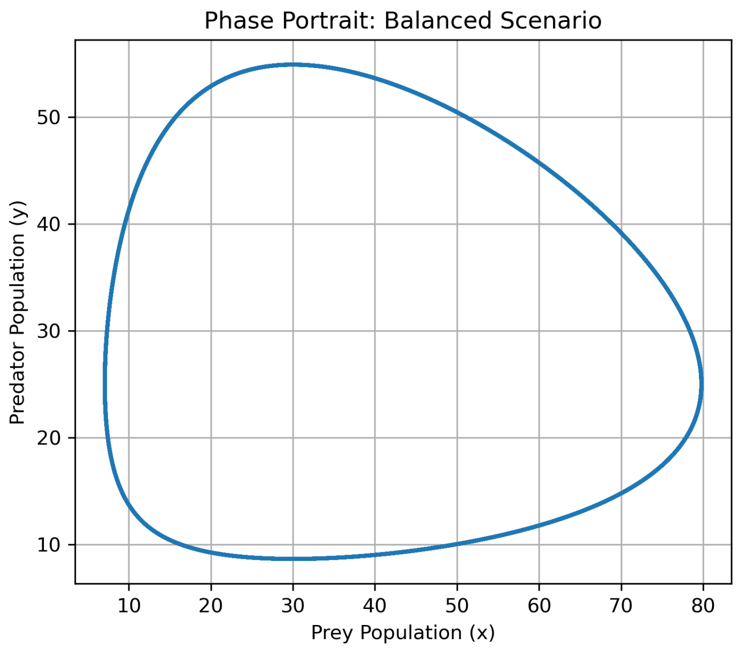

Figure 3: Phase portrait for the balanced initial conditions ($x(0) = 40$, $y(0) = 9$). The closed-loop trajectory illustrates sustained, periodic predator-prey oscillations.

Low Predator Initial Conditions: Phase Portrait

This simulation explores the behavior of the predator-prey system when predators begin at a very low population level. The initial conditions are defined as $x(0) = 50$ for the prey and $y(0) = 2$ for the predators. The parameters of the system remain consistent with earlier simulations: $\alpha = 0.5$, $\beta = 0.02$, $\delta = 0.01$, and $\gamma = 0.3$.

In this scenario, the prey population is initially high, while the predator population is near the threshold of extinction. Despite this imbalance, the system does not collapse. Instead, predators benefit from the plentiful prey and recover over time. As the predator population rebounds, the system gradually evolves into a stable, closed-loop trajectory in the phase space.

The shape of the resulting trajectory is similar to that of the balanced case but has a slightly

larger amplitude, reflecting the greater initial disparity between the species. Importantly, no extinction is observed for either population, and the system successfully returns to a sustainable cycle.
This case offers valuable insight into the system's resilience. Despite starting near predator extinction, the population dynamics demonstrate that the model naturally drives the system back into a self-sustaining oscillatory regime. This resilience is characteristic of the Lotka–Volterra system when both species are present, even if one begins in very small numbers.
The behavior observed in this scenario also addresses the project's guiding questions effectively. Although the predator population starts at a very low level, it recovers over time due to the high availability of prey. The populations evolve into a closed-loop cycle, with predator and prey numbers rising and falling in a stable pattern. Eventually, the system settles into a periodic oscillation similar to the balanced case, albeit with slightly altered amplitude. Despite the initial imbalance, the simulation does not show any signs of critical failure or collapse, reinforcing the model's inherent resilience and stability under low predator conditions.

```
# Low predator scenario: x0 = 50, y0 = 2
t_lp, x_lp, y_lp = simulate_predator_prey(50, 2, 200, 0.05)

plt.figure(figsize=(6, 5))
plt.plot(x_lp, y_lp, lw=2)
plt.title("Phase Portrait: Low Predator Scenario")
plt.xlabel("Prey Population (x)")
plt.ylabel("Predator Population (y)")
plt.grid(True)
plt.savefig("phase_portrait_low_predator.png", dpi=300, bbox_inches='tight
    ')
plt.show()
```

Listing 4: Phase portrait simulation for low predator initial conditions

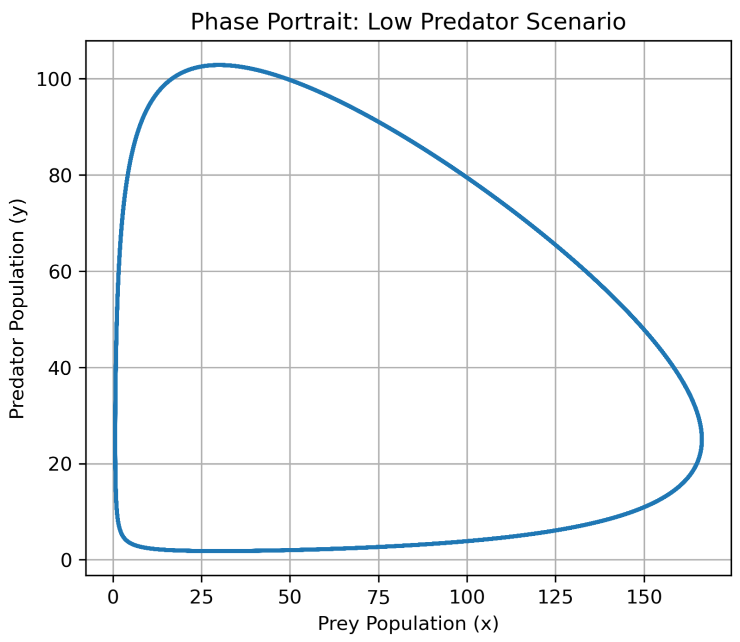

Figure 4: Phase portrait for the low predator initial conditions ($x(0) = 50$, $y(0) = 2$). The trajectory demonstrates predator recovery and emergence of a stable limit cycle despite initial imbalance.

High Predator Initial Conditions: Phase Portrait

This simulation investigates the behavior of the predator-prey system when the initial predator population is very high, and the prey population is comparatively low. The initial conditions are set as $x(0) = 20$ and $y(0) = 25$, with parameters $\alpha = 0.5$, $\beta = 0.02$, $\delta = 0.01$, and $\gamma = 0.3$. In this configuration, predators are initially overabundant relative to the available prey. As a result, the predator population consumes the limited prey supply at a rapid rate. This leads to a sharp decline in predator numbers due to starvation. However, the prey population eventually recovers in the absence of excessive predation, followed by a gradual rebound in the predator population as food becomes more abundant again. The long-term outcome is a closed-loop trajectory in phase space, similar in shape to that observed in the balanced scenario, though initially more volatile and phase-shifted.

This result demonstrates that the system is capable of recovering from extreme initial imbalances. Although the populations undergo significant stress early in the simulation, they ultimately stabilize into a sustainable oscillatory cycle.

With respect to the project's central questions, this scenario provides several important insights. First, it demonstrates how the populations evolve over time: although the prey is heavily suppressed in the early stages due to the initially high predator population, both species eventually recover and settle into a stable dynamic. Second, the system ultimately returns to periodic oscillations, indicating that the imbalance does not drive the populations to extinction or equilibrium. Finally, despite initial stress and resource scarcity, the system does not cross any critical thresholds leading to collapse or explosion. Instead, it exhibits resilience and re-establishes sustained cycles, thereby confirming the robustness of the predator-prey dynamics even when the initial conditions are heavily skewed in favor of one species.

```
# High predator scenario: x0 = 20, y0 = 25
t_hp, x_hp, y_hp = simulate_predator_prey(20, 25, 200, 0.05)
```

```
plt.figure(figsize=(6, 5))
plt.plot(x_hp, y_hp, lw=2)
plt.title("Phase Portrait: High Predator Scenario")
plt.xlabel("Prey Population (x)")
plt.ylabel("Predator Population (y)")
plt.grid(True)
plt.savefig("phase_portrait_high_predator.png", dpi=300, bbox_inches='
    tight')
plt.show()
```

Listing 5: Phase portrait simulation for high predator initial conditions

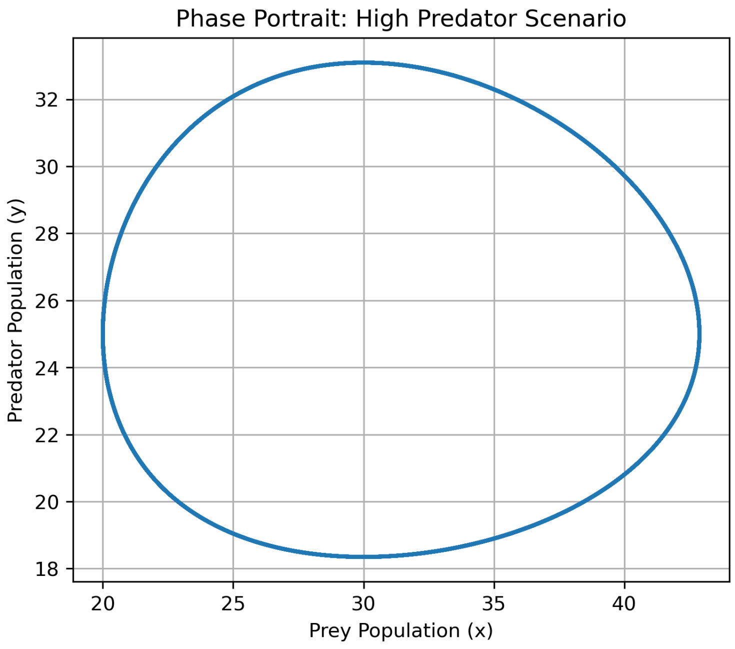

Figure 5: Phase portrait for the high predator initial conditions ($x(0) = 20$, $y(0) = 25$). The trajectory illustrates predator overconsumption and recovery, ultimately settling into a periodic cycle.

Near-Equilibrium Initial Conditions: Trajectories

This section investigates the behavior of the predator-prey system when the initial conditions are slightly perturbed from the analytically derived equilibrium point. The equilibrium occurs at $x^* = \frac{\gamma}{\delta} = 30$ and $y^* = \frac{\alpha}{\beta} = 25$, using the system parameters $\alpha = 0.5$, $\beta = 0.02$, $\delta = 0.01$, and $\gamma = 0.3$.

In this analysis, several trajectories are simulated beginning with initial conditions slightly offset from the equilibrium state. For example, perturbations such as $(x_0, y_0) = (29, 24)$ and $(31, 26)$ are used to assess the system's response to minor deviations from the steady-state.

The resulting figure illustrates that all trajectories form smooth, closed loops centered around the equilibrium point. Notably, none of the trajectories spiral inward, which would be indicative of a damped system, nor do they spiral outward, as would occur in an unstable or divergent system. Instead, the behavior confirms that the equilibrium point functions as a neutrally stable center.

This behavior is consistent with the nature of the Lotka–Volterra model, which is a conservative system and does not include any damping or energy dissipation terms. As such, solutions that begin near the fixed point tend to orbit indefinitely without converging or diverging. The presence of these stable, closed orbits under small perturbations reinforces the model's characteristic sensitivity to initial conditions near the equilibrium.

With respect to the project's guiding questions, the near-equilibrium scenario provides valuable confirmation of theoretical expectations. When the system is initialized at or near the fixed point $(30, 25)$, the populations remain bounded and exhibit stable, cyclical behavior. This demonstrates that the system does not evolve toward extinction or uncontrolled growth. Instead, the dynamics remain confined to smooth periodic orbits around the equilibrium. These results verify that the equilibrium state of the Lotka–Volterra system is neutrally stable: it neither attracts nor repels nearby trajectories. Additionally, the system exhibits persistent oscillations rather than convergence or divergence, reaffirming the absence of a critical parameter threshold in the vicinity of the fixed point. This scenario thus confirms that even near-perfect balance leads to sustained cyclic dynamics, consistent with a conservative nonlinear system.

In conclusion, this simulation visually confirms that the point $(30, 25)$ is surrounded by a family of closed, periodic orbits. Any small displacement from equilibrium results in sustained, bounded oscillations — fully aligned with classical Lotka–Volterra dynamics and the theoretical structure of the model.

```
# Generate trajectories near the equilibrium point
equilibrium_x = 0.3 / 0.01   # gamma / delta = 30
equilibrium_y = 0.5 / 0.02   # alpha / beta = 25

# Slight perturbations around equilibrium
offsets = [(-1, -1), (-1, 1), (1, -1), (1, 1), (0.5, 0.5), (-0.5, -0.5)
    ]
near_eq_trajectories = []

for dx, dy in offsets:
    x0 = equilibrium_x + dx
    y0 = equilibrium_y + dy
    t_vals, x_vals, y_vals = simulate_predator_prey(x0, y0, 200, 0.05)
    near_eq_trajectories.append((x_vals, y_vals, f"$x_0={x0}, y_0={y0}$
        "))

# Plot all perturbed trajectories around equilibrium
plt.figure(figsize=(7, 6))
for x_vals, y_vals, label in near_eq_trajectories:
    plt.plot(x_vals, y_vals, label=label)

plt.plot(equilibrium_x, equilibrium_y, 'ko', label='Equilibrium Point
    (30, 25)')
plt.title("Trajectories Near Equilibrium Point")
plt.xlabel("Prey Population (x)")
plt.ylabel("Predator Population (y)")
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5), title="Initial
     Conditions")
plt.grid(True)
plt.savefig("trajectories_near_equilibrium.png", dpi=300, bbox_inches='
    tight')
plt.show()
```

Listing 6: Simulating perturbed trajectories near the equilibrium point

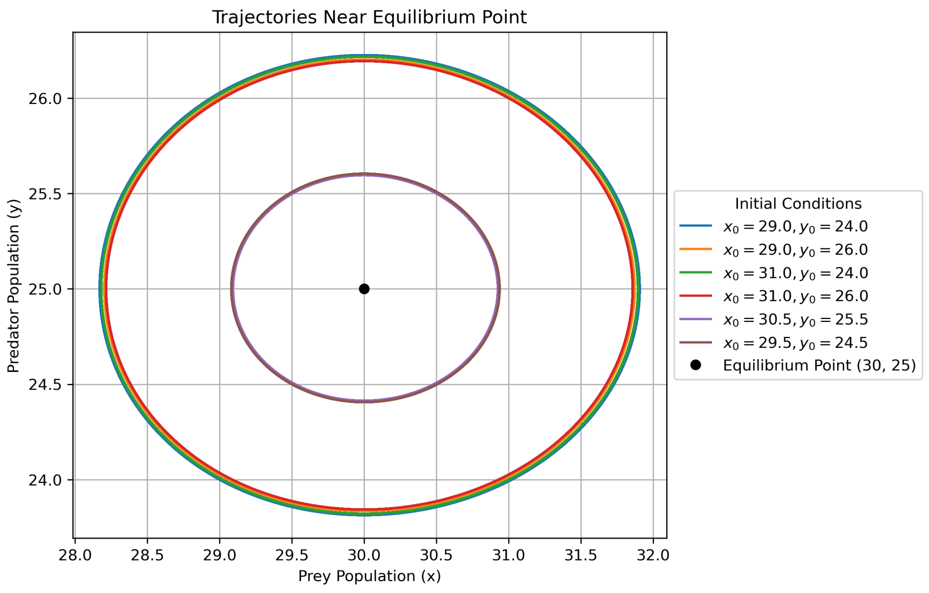

Figure 6: Phase portrait of trajectories beginning near the equilibrium point $(30, 25)$. Each closed orbit corresponds to a slightly perturbed initial condition, confirming the neutral stability of the equilibrium.

Zero Predator Initial Conditions: Phase Portrait

This final scenario explores an edge case in which predators are entirely absent at the start of the simulation. The system is initialized with $x(0) = 50$ and $y(0) = 0$, using the same parameters as in previous simulations: $\alpha = 0.5$, $\beta = 0.02$, $\delta = 0.01$, and $\gamma = 0.3$.

In this situation, there are no predators to suppress the prey population. As a result, the prey population is unconstrained and grows exponentially over time. Since predator reproduction depends on interaction with prey, and the predator population starts at zero, it remains at zero throughout the simulation. This creates a one-sided dynamic in which only the prey population changes, with no feedback loop to regulate its growth.

The phase portrait illustrates this breakdown in interaction. The predator population remains fixed at $y = 0$, and the trajectory forms a flat curve along the x-axis. This reflects a total collapse in predator dynamics, as the system becomes decoupled and behaves like a single-species exponential growth model.

With respect to the original questions posed in this project, this case yields clear and significant conclusions. The predator population immediately goes extinct and remains at zero, while the prey population explodes in size without any regulating force. No oscillatory behavior or equilibrium is observed; instead, the system diverges into an imbalanced and unsustainable state. This scenario highlights the importance of predator presence: their complete absence represents a critical condition that leads directly to collapse. As such, this case confirms that certain threshold conditions—such as zero initial predator population—can irreversibly destabilize the system.

```
# Zero predator scenario: x0 = 50, y0 = 0
t_zp, x_zp, y_zp = simulate_predator_prey(50, 0, 200, 0.05)

plt.figure(figsize=(6, 5))
plt.plot(x_zp, y_zp, lw=2)
plt.title("Phase Portrait: Zero Predator Scenario")
plt.xlabel("Prey Population (x)")
plt.ylabel("Predator Population (y)")
plt.grid(True)
plt.savefig("phase_portrait_zero_predator.png", dpi=300, bbox_inches='
    tight')
plt.show()
```

Listing 7: Phase portrait simulation for zero predator initial conditions

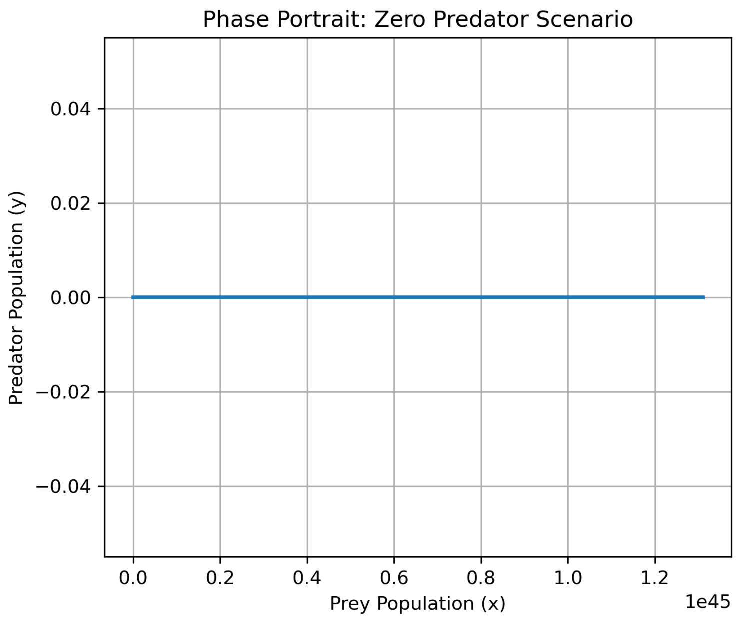

Figure 7: Phase portrait for the zero predator initial conditions ($x(0) = 50$, $y(0) = 0$). The predator population remains extinct, while the prey grows unchecked. The trajectory forms a flat line along the x-axis, indicating a complete breakdown of predator-prey interaction.

Summary of Predator-Prey Simulation Results Across All Scenarios

This section synthesizes the key insights gained from the five phase portraits, each of which corresponds to a different initial condition. Together, these simulations provide a thorough answer to the three central questions outlined at the beginning of this study.

1. Balanced Initial Conditions

The system is initialized with $x(0) = 40$ and $y(0) = 9$. The resulting phase portrait shows a smooth, symmetric closed-loop trajectory, where both prey and predator populations remain bounded and oscillate indefinitely. This simulation illustrates the textbook behavior of a predator-prey system governed by the Lotka–Volterra model. The populations grow and decline in a repeating pattern, with no signs of collapse, equilibrium, or runaway growth.

2. Low Predator Initial Conditions

For this simulation, the initial conditions are $x(0) = 50$ and $y(0) = 2$, representing a scenario with abundant prey and a near-extinct predator population. Initially, the trajectory reflects this imbalance, but over time it stabilizes into a closed loop with a larger amplitude than the balanced case. The predator population recovers as it feeds on the plentiful prey, ultimately forming a self-sustaining cycle. This scenario highlights the system's ability to rebound from extreme predator scarcity and demonstrates the robustness of the model to initial population imbalance.

3. High Predator Initial Conditions

Here, the system begins with $x(0) = 20$ and $y(0) = 25$, representing an environment with relatively few prey and an overabundant predator population. In the early stages, the predators overconsume the limited prey supply, causing their own population to crash. However, once predator numbers decline, the prey population recovers, and the predator population follows. The trajectory eventually forms a closed orbit, confirming that the system does not collapse even under extreme initial conditions. The long-term behavior remains periodic and bounded.

4. Near-Equilibrium Initial Conditions

In this scenario, the system is initialized very close to the analytically derived equilibrium point: $x(0) = 30$, $y(0) = 25$. The resulting phase portrait displays a very small loop around the equilibrium, caused by minor numerical perturbations. The system does not drift away from this point, confirming that the equilibrium is neutrally stable. In the absence of rounding error, the populations would remain perfectly fixed. However, due to the numerical implementation, nearby trajectories form small, stable orbits, validating the expected behavior near the fixed point.

5. Zero Predator Initial Conditions

This edge case explores the scenario in which predators are completely absent at the start: $x(0) = 50$, $y(0) = 0$. Without any predators to regulate growth, the prey population increases without bound. The predator population remains extinct due to a lack of reproductive interaction. The resulting phase portrait is a flat trajectory along the x-axis, indicating a complete collapse of predator dynamics. This is the only scenario that leads to a breakdown in system balance and highlights the necessity of both populations for ecological stability.

How These Results Answer Our Key Questions

1. How do the predator and prey populations evolve over time?

In Scenarios 1 through 4, the predator and prey populations consistently evolve into periodic oscillations after an initial adjustment phase. The system does not settle at a fixed point unless initialized exactly at the equilibrium, and even then, minor perturbations cause it to cycle. In contrast, the zero-predator scenario leads to unbounded prey growth and predator extinction, marking a significant departure from the typical cyclic behavior.

2. Do the populations exhibit periodic oscillations, reach equilibrium, or experience extinction?

Periodic oscillations are observed in Scenarios 1 through 4, affirming the core dynamics of the Lotka–Volterra model. In Scenario 4, where the system starts exactly at the fixed point, any deviation due to rounding errors results in small, closed loops—further confirming the neutral stability of the equilibrium. Scenario 5 is unique in that it leads to extinction of the predator species and exponential prey growth, with no oscillation or equilibrium, making it a clear example of system collapse.

3. Are there critical parameter values or initial conditions that lead to collapse or explosion?

Yes, Scenario 5 clearly demonstrates that starting with zero predators, $y(0) = 0$, causes an irreversible collapse in the predator population and leads to unchecked exponential growth in the prey population. This case shows that while the system is robust to variations in initial conditions, it is not resilient to the total absence of one species. The presence of both predator and prey populations is essential for maintaining bounded, periodic behavior.

Conclusion

The Lotka–Volterra predator-prey system displays a wide range of nonlinear behaviors depending on initial conditions. In most cases, the system is resilient: even when initialized with severe imbalance, it returns to periodic, bounded oscillations. However, a critical threshold does exist—when one species is entirely absent, the system collapses. These findings underscore both the stability and fragility of predator-prey dynamics and confirm the RK4 method as an effective numerical tool for exploring long-term population behavior.

Section 5: Group Summary

Our group had an even distribution of work throughout this project. Each member helped complete each section of the report, with some members focusing their efforts on specific portions and others taking the lead in different areas. This summary outlines each group member's personal contributions to the project.

Ayush

Portion of work: ~35%
Ayush made significant contributions to Section 1 and was instrumental in creating the question set to be addressed throughout the report, as well as determining the initial conditions to be tested using our numerical method. His work on the initial conditions also translated to substantial contributions in Section 4, analyzing the results of the system for each case. Additionally, he supported work across all other sections and assisted in formatting the final report.

Hiko

Portion of work: ~30%
Hiko took on the majority of the work for Section 2, detailing the purpose, justification, and mathematical formulation of the Runge-Kutta 4th-order (RK4) method used throughout the project. He also contributed to the overall formatting of the report and supported calculations and explanations in other sections as needed.

Benjamin

Portion of work: ~35%
Benjamin led the creation of plots related to the numerical method, error convergence, and system results across Sections 3 and 4. He collaborated with Ayush on determining the question set and initial conditions. Beyond his lead responsibilities, Benjamin also supported other members with explanations and implementation of various parts of the project.

Group Collaboration

As a whole, our group worked effectively and collaboratively. Communication was clear and efficient, allowing us to stay on track for project completion. Each member made meaningful contributions and supported one another, resulting in a cohesive and comprehensive final report that reflects our shared efforts and understanding of the topic.

Section 6: Reproduciability

The code for this project is available at the GitHub link: https://github.com/ayushpathy/AE370$_{P}roject$1

Acknowledgements

This report used ChatGPT extensively throughout the project. Our group used ChatGPT to help us produce code, filter through our group's ideas to better fit the project prompt, and help us apply AE370 content to our report ideas. We provided the direction of this project. Our group decided to use the project to analyze the predator-prey dynamical system, and we also decided which questions to answer and how to answer them with mathematical results. ChatGPT aided the execution of these goals both in discerning what paths would be best to properly fufill the project guidelines and in performing the tasks themselves. ChatGPT helped with formatting the code in our .ipynb file and for this report, as well.