COM3001 Assignment

Aryan Golbaghi
Department of Computer Science, University of Sheffield
Sheffield, England
agolbaghi1@sheffield.ac.uk

April 15, 2025

Chapter 1

Question 1: Dynamical systems

1.1 Exercise 1: Lorenz dynamical system

The Lorenz system is a set of three coupled, nonlinear differential equations [15], which was introduced by Edward Lorenz in 1963 as a simplified model for atmospheric convection. Originally it was derived to study weather prediction and its formulation is:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z \tag{1.1}$$

$$\text{where } \sigma = 10, \ \rho = 28, \text{ and } \beta = \tfrac{8}{3}.$$

It revealed that even deterministic systems can exhibit unpredictable and chaotic behavior. Lorenz's discovery of the sensitive dependence on initial conditions is now known as the "butterfly effect" [17]: a concept that demonstrates how small changes in the initial state of a system can lead to drastically divergent trajectories over time, and it has transformed the understanding of dynamical systems by showing that small differences in starting states can lead to dramatically divergent results over time.

Historically, the Lorenz system marked a paradigm shift. Before Lorenz's work, many scientists assumed that small changes in initial conditions would only lead to small variations in future behavior. However, Lorenz showed that the weather, which was considered a definitive example of predictable physical systems, could be inherently unpredictable. This insight laid the foundation for modern chaos theory and encouraged extensive research into nonlinear dynamics [17].

Beyond its initial role in meteorology, the Lorenz system has found widespread application in various fields. Its conceptual framework has influenced research in fluid dynamics, laser physics, electrical circuits, and even models of population dynamics [14]. In many of these areas, the Lorenz equations serve as a benchmark for studying chaotic attractors, strange attractors, and fractal dimensions, concepts that have proven essential in describing complex and real-world phenomena. The model's ability to capture the essence of chaotic behavior makes it invaluable not only as a theoretical tool but also as a practical one for understanding and predicting complex systems.

In recent years, the relevance of the Lorenz system has grown further with advances in computational methods and interdisciplinary applications. For instance, researchers have employed data assimilation techniques and machine learning approaches such as reservoir computing to both forecast and analyze chaotic dynamics using the Lorenz attractor as a prototype. These modern developments underscore the system's enduring importance: they not only provide new tools for dealing with uncertainties in weather forecasting but also offer insights into controlling chaotic systems in engineering and natural sciences [16]. Furthermore, studies exploring quantum computing approaches to analyze the non-Hermitian Jacobian matrix of the Lorenz system have opened promising avenues for future research in atmospheric physics and beyond [2].

Overall, the Lorenz system remains very important in the study of nonlinear dynamical systems. Its historical impact has been reshaping our understanding of predictability and determinism, and it continues to influence modern scientific research. As contemporary studies extend its applications into new fields, the Lorenz model exemplifies how a deceptively simple set of equations can encapsulate the profound behavior of nature, making it directly relevant to a wide array of exercises and research endeavors in mathematical modeling.

1.1.1 How to use numerical methods to simulate the system,

As shown in the Lorenz Equation 1.1, Because of the system's nonlinearity and sensitivity to initial conditions (the "butterfly effect"), an analytical solution is not feasible. Instead, we can use numerical integration. In this section, I present two methods, alongside an analysis of time step effects and the simulation results:

The Explicit Euler Method

The Euler method is a first order numerical scheme. For a general differential equation

$$\frac{dx}{dt} = f(x),$$

the Euler update rule is for Lorenz System is:

$$x_{n+1} = x_n + \Delta t\,\sigma(y_n - x_n), \quad y_{n+1} = y_n + \Delta t\,[x_n(\rho - z_n) - y_n], \quad z_{n+1} = z_n + \Delta t\,[x_n y_n - \beta z_n] \tag{1.2}$$

where Δt is the time step.

Although this formula is straightforward, the Euler method is only first order accurate; and its error is proportional to Δt. For the Lorenz system 1.1, this may lead to significant inaccuracies when using larger time steps, especially due to the chaotic nature of the system.

The Fourth-Order Runge-Kutta (RK4) Method

RK4 is a more accurate method that uses intermediate calculations to achieve fourth-order accuracy. The RK4 update rule for the Lorenz system is given by:

$$\begin{aligned}
k_1 &= f(x_n, y_n, z_n), \\
k_2 &= f\left(x_n + \tfrac{1}{2}\Delta t\,k_{1,x},\ y_n + \tfrac{1}{2}\Delta t\,k_{1,y},\ z_n + \tfrac{1}{2}\Delta t\,k_{1,z}\right), \\
k_3 &= f\left(x_n + \tfrac{1}{2}\Delta t\,k_{2,x},\ y_n + \tfrac{1}{2}\Delta t\,k_{2,y},\ z_n + \tfrac{1}{2}\Delta t\,k_{2,z}\right), \\
k_4 &= f\left(x_n + \Delta t\,k_{3,x},\ y_n + \Delta t\,k_{3,y},\ z_n + \Delta t\,k_{3,z}\right), \\
x_{n+1} &= x_n + \frac{\Delta t}{6}\left(k_{1,x} + 2k_{2,x} + 2k_{3,x} + k_{4,x}\right), \\
y_{n+1} &= y_n + \frac{\Delta t}{6}\left(k_{1,y} + 2k_{2,y} + 2k_{3,y} + k_{4,y}\right), \\
z_{n+1} &= z_n + \frac{\Delta t}{6}\left(k_{1,z} + 2k_{2,z} + 2k_{3,z} + k_{4,z}\right)
\end{aligned} \tag{1.3}$$

Because of its higher accuracy, RK4 is better suited for systems like the Lorenz, where small errors can lead to significant divergence.

Time Step Analysis

Choosing an appropriate Δt is crucial:

- **Smaller** Δt (e.g., 0.01, 0.001) improves accuracy by capturing more details of the system's dynamics but increases computation expenses.
- **Larger** Δt (e.g., 0.05, 0.08) may be computationally cheaper but can lead to numerical instability and incorrect behavior in the simulation.

While fixed time step methods such as Euler and RK4 are usefull, adaptive integration methods like Runge-Kutta-Fehlberg (RK45) can automatically adjust Δt based on local error estimates. This adaptive control is particularly useful in chaotic systems, where the dynamics can vary significantly over time, ensuring both efficiency and robustness in long-term simulations. By dynamically allocating computational resources, reducing the step size in regions of rapid change and increasing it in smoother regions, adaptive methods not only maintain accuracy but also improve performance.

Simulation and Results

The simulations were conducted using both the Explicit Euler and the Fourth-Order Runge-Kutta (RK4) methods over a total simulation time of 40 seconds [5]. Several time step sizes were considered, ranging from small steps ($\Delta t = 0.001$) up to larger steps ($\Delta t = 0.1$). The results of the simulations are presented in the following figures 2.1, 2.2:

My observations for different time steps are as follows: **Small** Δt (e.g., 0.001): Both methods yield a stable, accurate Lorenz attractor. RK4 shows less numerical distortion than Euler.

Intermediate Δt (0.01): Euler solutions begin to show noticeable distortion. RK4 typically remains close to the true attractor, but still accumulates some error.

Large Δt (0.1): Euler diverges dramatically. RK4 diverges from the solution too, however the RK4 method stays more loyal to the true solution.

In figure 2.2 shows that the Euler method is conditionally stable, meaning that for chaotic systems like the Lorenz system, even small time steps might lead to instability over long simulations, however the RK4 method has a larger stability region, which is why it can both stay stable and performs better at larger time steps compared to Euler.

Quantitative Error Analysis

To assess the performance of the numerical methods, we compare the computed trajectories to a high-accuracy reference solution obtained with the RK4 method at a very small time step ($\Delta t = 0.001$). In this context, the accuracy of a method is quantified by the *average Euclidean error*, defined as:

$$\text{Error} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{X}_i^{\text{method}} - \mathbf{X}_i^{\text{ref}} \right\|,$$

where $\mathbf{X}_i = (x_i, y_i, z_i)$ represents the state vector at the ith time step, $\mathbf{X}_i^{\text{method}}$ is the solution produced by the numerical method being tested, and $\mathbf{X}_i^{\text{ref}}$ is the corresponding state from the reference trajectory.

Theoretical Error Scaling: It is well known that different numerical integration methods exhibit distinct error behaviors:

- **Euler Method:** The Euler method has a *local truncation error* on the order of $O(\Delta t^2)$. However, because errors accumulate over $T/\Delta t$ steps, the *global error* scales as $O(\Delta t)$ (i.e., linearly with the time step). This implies that even small increases in Δt will lead to a proportional increase in the total error.
- **Fourth-Order Runge-Kutta (RK4) Method:** In contrast, the RK4 method exhibits a much smaller local truncation error of $O(\Delta t^5)$, resulting in a global error that scales as $O(\Delta t^4)$. This steep reduction in error with decreasing Δt makes RK4 far more accurate for the same step sizes.

The simulation results align with these theoretical expectations. For example, as seen in Figure **??**, when the time step increases beyond 0.01, the Euler method's error grows rapidly, failing to accurately reproduce the Lorenz attractor. In contrast, the RK4 method maintains a significantly lower error, demonstrating its superior stability and accuracy even at moderately larger time steps.

Error Metric Details: The average Euclidean error is computed by comparing the solution at each time step from the numerical method under investigation with the corresponding state of the reference solution. By averaging the Euclidean distances over all time steps, we obtain a single scalar measure of the overall deviation. This metric provides a straightforward yet effective means to quantify how numerical errors accumulate over time, particularly in chaotic systems where small errors can lead to substantial deviations in the trajectory.

Overall, these analyses not only confirm the expected theoretical error scaling for each method but also emphasize the critical importance of choosing appropriate time steps when simulating sensitive, nonlinear systems such as the Lorenz attractor.

1.2 Exercise 2: Selecting a Numerical Method and Demonstrating Chaotic Behavior

Selection of Numerical Method and Time Step

Based on the analyses in Exercises 1.2.4 and 1.2.5, I have chosen the Fourth-Order Runge-Kutta (RK4) method for simulating the Lorenz system. The choice of the RK4 method is supported by the following considerations:

- **Accuracy:** As illustrated in Figure 2.3 and Table 2.1, the RK4 method exhibits substantially lower local truncation errors compared to the Euler method, making it more reliable in capturing the intricate dynamics of the system.
- **Stability:** Experimental results (see Figure 2.4) indicate that the Euler method becomes unstable for time steps exceeding $\Delta t > 0.025$. In contrast, RK4 remains stable even for time steps as large as $\Delta t = 0.13$ (see 2.5), proving its robustness for simulating chaotic systems.

- **Computational Efficiency:** Although smaller time steps can improve accuracy, they also lead to higher computational costs. To strike an appropriate balance between accuracy and efficiency, I employed a hyperparameter optimization approach using the Optuna library [1] (see also my implementation in [6]). A sensitivity analysis was conducted by varying Δt using the objective function (1.4). This analysis confirmed that a time step of $\Delta t = 0.0073$ (see Figure 2.6 for comparison).

Optimization was guided by a **composite objective function**, which incorporates both the simulation error and a proxy for computational cost:

$$\text{Composite Objective} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_{\text{sim}}(t_i) - \mathbf{x}_{\text{ref}}(t_i)\|_2 + \alpha \cdot \left(\frac{T}{\Delta t}\right) \tag{1.4}$$

where: $\mathbf{x}_{\text{sim}}(t_i)$ is the simulated state at time t_i,
$\mathbf{x}_{\text{ref}}(t_i)$ is the reference state at time t_i,
N is the number of time steps in the simulation,
$\|\cdot\|_2$ denotes the Euclidean (L2) norm,
α is a weighting factor for the cost term, with $\alpha = 0.0009$ in this case,
T is the total simulation time,
Δt is the time step used.

This formulation encourages the optimizer to select a time step that balances simulation accuracy with computational efficiency, avoiding excessively fine resolutions. However, its performance may depend on the specific weighting parameter α (manually tuned here) and the reference system used.

Generation and Analysis of Phase Portraits

To demonstrate the characteristic chaotic behavior ("butterfly effect") of the Lorenz system, I have chosen two similar initial conditions of (0.01, 2.0, 1.0) and (0.1, 2.0, 1.0) and the time steps of $\Delta t = 0.0073$.

Figures 2.7, 2.8 and 2.9 display these phase portraits. Although the initial conditions differ only in the x component by a small amount, the resulting trajectories clearly exhibit chaotic divergence over time, Both portraits exhibit the same broad "butterfly" structure, but their trajectories do not remain synchronized. As time goes on, the small difference amplifies until the solutions appear uncorrelated.

1.2.1 What Does It Mean for a System to Be Chaotic?

A chaotic system is a deterministic system that is unpredictable and seemingly random behavior due to its extreme sensitivity to initial conditions. This means that even very small differences in starting conditions can lead to completely different outcomes over time. Such systems are characterized by aperiodic long-term behavior, meaning they do not settle into regular cycles, and their evolution over time appears disorderly despite being governed by deterministic laws.

for more details on variances in in the initial condition more details can be found in theb **Generation and Analysis of Phase Portraits** section.

Chapter 2

Question 2: Prey-Predator System

2.1 Simulate the prey-predator model

The result of the prey-predator simulation can be found at Figures (2.10, 2.11), here is a description of the simulation:

In the simulation, we can see an oscillatory behavior of the populations, which is a characteristic feature of predator-prey dynamics.

- Grass (black curve) grows, fueling an increase in Rabbits (orange).
- As rabbits grow numbers, Foxes (blue) find abundant prey, so fox numbers grow.
- Increasing the number of rabbits will cause to decrease the amount of grass (this is seen in Figure 2.11 too).
- Once foxes grow in numbers and the grass amount decreases because of the population of rabbits, the population of rabbits will decrease.
- As the number of rabbits decreases, the foxes will have less food, leading to a decrease in their population.
- as there is less rabbits, the grass will grow again, and the cycle will repeat.

The intreating findings I can see from this simulation is even though the agent based model includes randomized movement and local interactions, the overall population still exhibits an oscillatory up and down cycles, similar to what we see in standard predator prey models (Lotka Volterra curves which mentioned in the lectures).

2.2 Differences between Agent Based Modeling vs Equation Based Modeling

In a classical equation based (ODE) predator prey model, such as Lotka Volterra equations:

$$\begin{cases} \dfrac{dx}{dt} = x(\alpha - \beta y) \\ \dfrac{dy}{dt} = y(\delta x - \gamma) \end{cases} \qquad \begin{aligned} &\alpha > 0, \quad \text{Reproduction prey} \\ &\beta > 0, \quad \text{Predation} \\ &\gamma > 0, \quad \text{Extinction predator} \\ &\delta > 0, \quad \text{Reproduction predator} \end{aligned}$$

populations are treated as continuous, homogeneous averages. By contrast, agent based models (ABMs) track individual rabbits and foxes. I have summarized the key differences between these two models in the following (see this table 2.3):

- **Spatial Heterogeneity**
  - **ABM:** Rabbits and foxes move around, so local depletion of grass or clustering of rabbits affects local outcomes. Some regions can be overgrazed or overhunted while others remain safe havens, and these local effects alter the overall population dynamics.

  - **ODE:** Assumes well mixed populations with no spatial structure. Grass, rabbits, and foxes interact uniformly in a single compartment.

- **Discrete and Stochastic Interactions**
  - **ABM:** Each eating or breeding event is discrete. Foxes may or may not successfully catch a rabbit (random probability), and rabbits may or may not find enough grass. This random element leads to fluctuations around the average trend.
  - **ODE:** Uses deterministic rate equations; there is no randomness in who gets eaten or how far you can move.

- **Individual Traits and Thresholds**
  - **ABM:** Each agent has individual properties such as an internal food store, an age, and breeding frequency. Once a rabbit's internal food is depleted, it dies—even if the average rabbit might have enough food.
  - **ODE:** Tracks the average rate of consumption and reproduction. There is no notion of an individual's internal energy or local shortage.

- **Emergent Behavior and Local Extinction**
  - **ABM:** Because movement is localized, pockets of rabbits or foxes can go extinct locally while others flourish elsewhere. Over time, random events can create drastically different patterns across multiple simulations.
  - **ODE:** Typically yields a single smooth trajectory for each initial condition—no chance of a local "pocket" dying off on its own.

- **Complexity vs. Analytical Tractability**
  - **ABM:** More realistic in capturing how individuals actually move, eat, and breed, but more complex and computationally intensive. Hard to get a neat "closed form" solution.
  - **ODE:** Mathematically elegant with known analysis techniques (e.g., stability analysis, limit cycles). Faster to run and simpler to interpret but less nuanced spatially and individually.

2.2.1 Why Do Results Differ Across Runs?

- **Random Initial Conditions**
  - Each run places rabbits and foxes at random locations in the grid.
  - Grass regeneration also occurs in randomly chosen cells.

- **Probabilistic Movement and Interactions**
  - Rabbits choose random directions if no grass is found in sight.
  - Foxes move randomly and only sometimes succeed in catching a rabbit (based on a probability that depends on the distance).
  - Each new rabbit or fox gets a random starting age, This affects how soon they die or reproduce.

2.2.2 Simulating the ABM Model to Monitor the Change of Key Variables

Here I have considered the following key variables, recorded across 100 simulation runs of the ABM. This information helps us evaluate the sustainability of an environment by revealing population dynamics and resource availability over time. Analyzing these trends can support better ecological understanding and inform strategies to prevent long-term or permanent environmental damage (please see Table 2.3 for the summary statistics of these variables and Figure **??** for their distributions).

- **Final Grass:** The total amount of grass available in the environment at the end of the full run of each simulation.
- **Final Rabbits:** The number of rabbits alive at the end of the full run of each simulation. This indicates if the rabbit population sustains itself or collapses over time.
- **Final Foxes:** The number of foxes alive at the end of the full run of each simulation. This helps determine whether the predator population can persist or dies out due to lack of prey.
- **Peak Grass:** The maximum amount of grass observed at any point during a simulation run.
- **Peak Rabbits:** The highest number of rabbits observed during a simulation run. This reflects the peak of reproduction and food availability.
- **Peak Foxes:** The maximum number of foxes observed during a simulation run. It shows the extent the predator population grows, in response to abundant prey.
- **Min Grass:** The lowest amount of grass recorded during the simulation. This indicates how depleted the resource can become due to overpopulation of rabbits.
- **Min Rabbits:** The minimum number of rabbits observed at any point. A value of zero suggests a population collapse or local extinction event.
- **Min Foxes:** The lowest number of foxes observed during a simulation. A minimum of zero suggests predator extinction, due to insufficient rabbit populations.

In Table 2.3, we observe that the system generally maintains ecological stability over the course of the simulations. The minimum amount of grass never reaches zero, which is a critical factor in ensuring the survival of the rabbit population, the rabbit population maintains a positive final count across all runs, showing that they are capable of reproducing and sustaining themselves.

However, due to the stochastic nature of agent-based models, we observe that in some simulation runs (6 out of 100), the fox population drops to zero. Despite this, the presence of surviving foxes in most simulations suggests the system can support predator-prey dynamics.

2.3 Parameter Sensitivity Analysis

In the previous exercise, we observed that the fox population went extinct in 6 out of 100 simulation runs. In this section, I aim to identify the conditions under which the fox population can be sustained more consistently. To do this, I selected a key parameter to vary and recorded the system's behavior in response.

The primary cause of fox extinction appears to be a shortage of rabbits, which in turn is often caused by a lack of grass and high number of foxes. Here I decided to vary the `growrate` parameter of the environment, which controls how many new grass units are added to the grid in each iteration. By increasing the grow rate, I aim to assess whether more abundant grass leads to larger rabbit populations, and subsequently, to better survival conditions for foxes. My first assumption is that a higher growth rate will result in a more stable ecosystem, allowing both rabbits and foxes to grow in mean population. However, I also predict the possibility of the opposite effect: the rabbit population might overgrow, leading to an increase in the fox population. As a result, the foxes could overhunt the rabbits, causing the rabbit population to crash. With no rabbits left to repopulate, the ecosystem could die.

if my first assumption is correct, I expect to see a higher mean in the population of the foxes however a similar standard deviation (A low standard deviation would indicate a a stable environment). However if my second assumption is correct, I expect to see a higher mean and higher standard deviation in the population of the foxes, indicating that they are grow initially in population due to abundance of the preys but due to overhunting, the foxes of the population will be at the risk of extinction.

I experimented with the following values of the growrate parameter: 60, 65, 70, 75, 80, 90, 100, 140, 180, and 220 to see their affects on the probability of foxes extinction and the average final population of the foxes. i ran 100 simulations for each values and 1000 iterations for each simulation. the results are shown the Figures

List of Figures

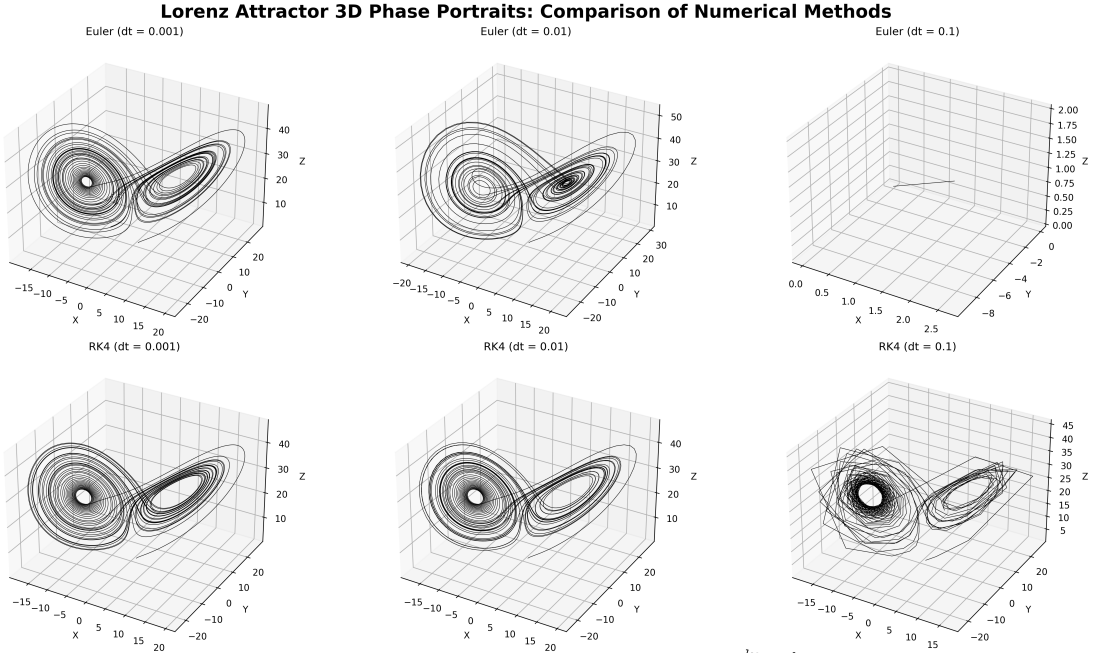

Figure 2.1: Lorenz attractor with initial state (1.0,1.0,1.0) and time step (0.001, 0.01, 0.1), and total simulation time of 40 seconds, For $\Delta t > 0.01$, the solution is not stable enough to reliably reproduce the attractor shape[7]

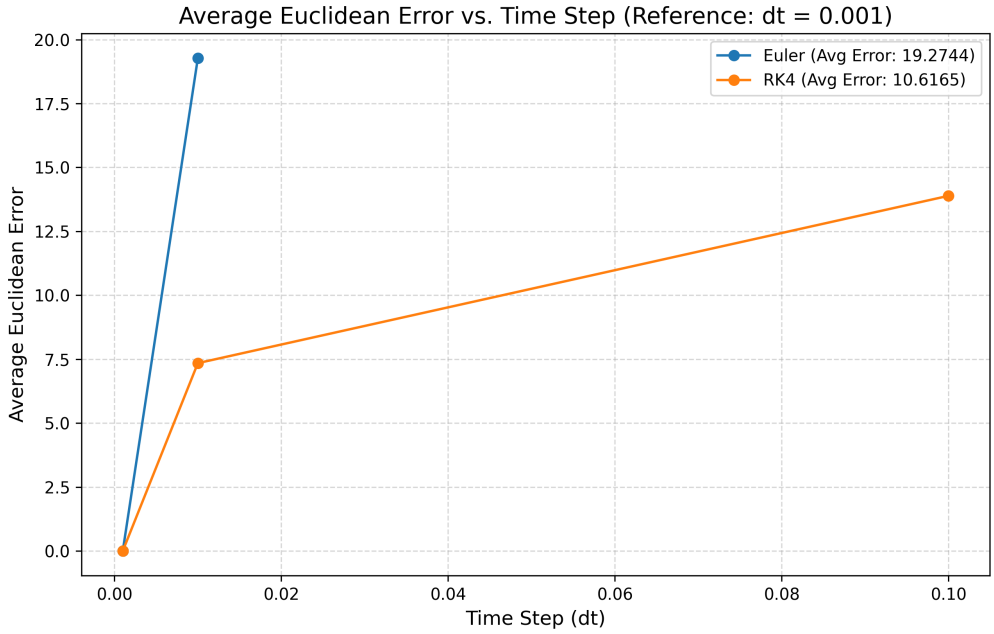

Figure 2.2: Lorenz attractor error analysis with initial state (1.0,1.0,1.0) and time step (0.001, 0.01, 0.1), and total simulation time of 40 seconds, For $\Delta t > 0.01$, the solution is not stable enough to reliably reproduce the attractor shape[7]

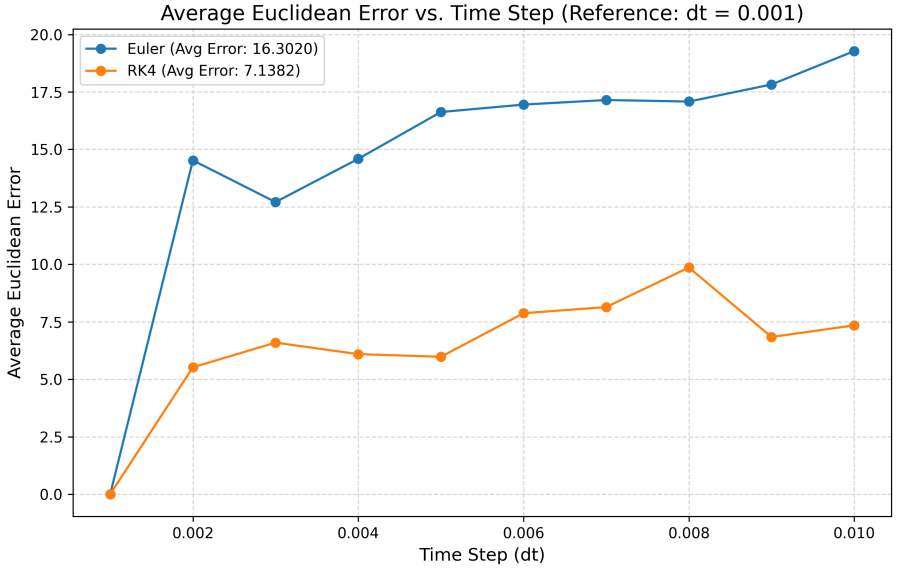

Figure 2.3: Lorenz attractor error analysis with initial state $(1.0, 1.0, 1.0)$, 1000 different time steps between $(0.0001 and 0.001)$, and a total simulation time of 40 seconds. this experiment is to compare the mean error of the two methods[8].

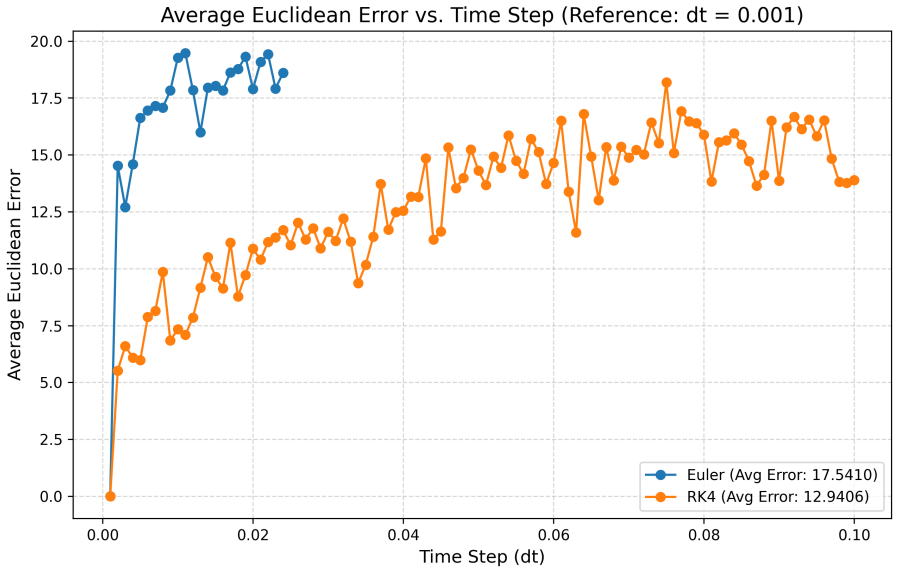

Figure 2.4: Lorenz attractor error analysis with initial state $(1.0, 1.0, 1.0)$, using 1000 different time steps between 10^{-3} and 10^{-1}, and a total simulation time of 40 seconds. For $\Delta t > 0.025$, the Euler solution is not stable enough to reliably reproduce the attractor shape. [4].

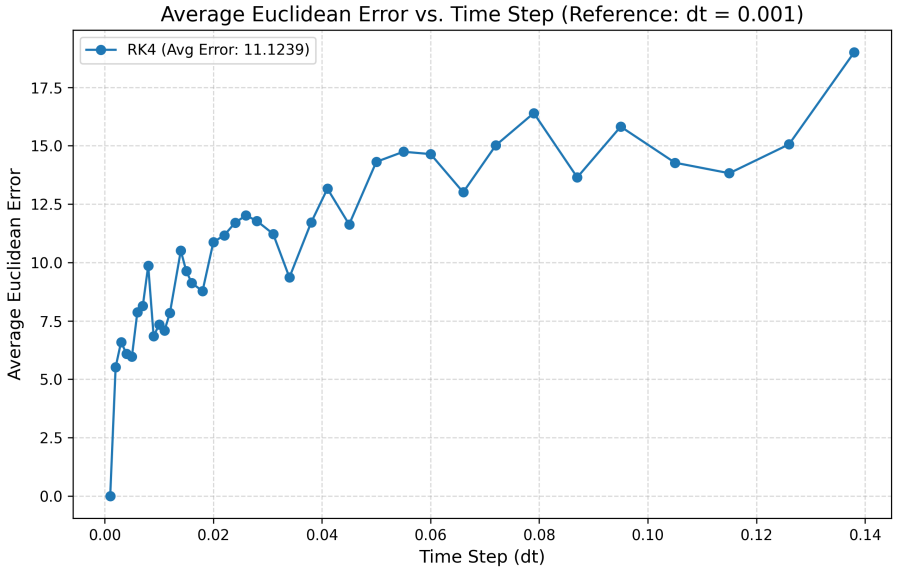

Figure 2.5: Lorenz attractor error analysis with initial state $(1.0, 1.0, 1.0)$, using 100 different time steps between 10^{-3} and 10^{1}, and a total simulation time of 40 seconds. For $\Delta t < 0.14$, the RK4 solution is stable enough to reproduce the attractor shape. [3].

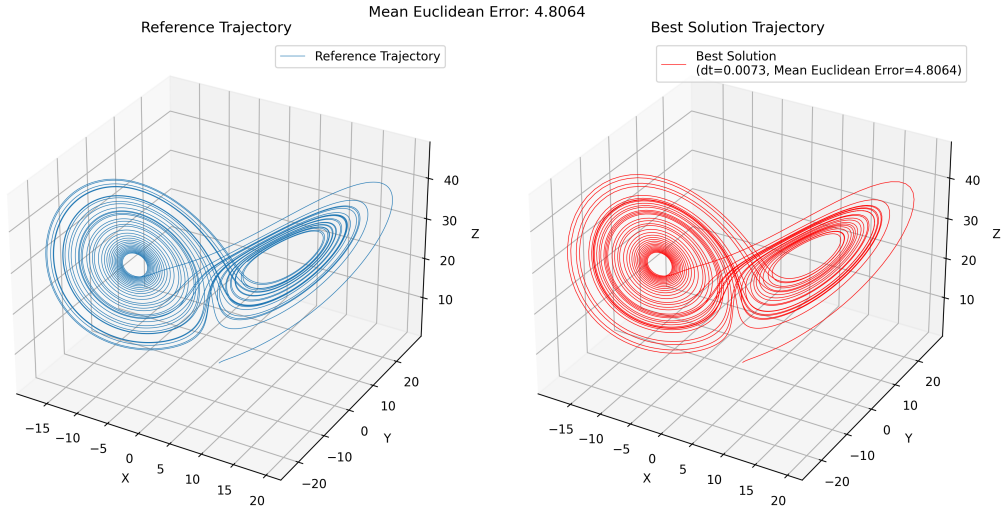

Figure 2.6: Comparison between reference system with $\Delta t = 0.001$ and a less computationally expensive system with step size of $\Delta t = 0.0073$, for this I used Optuna [1] with 50 trials. the code is at [6] and the image [9].

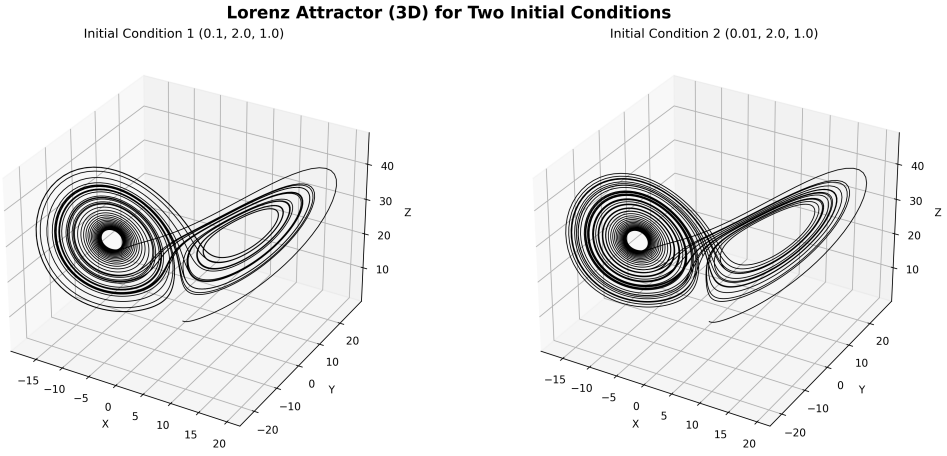

Figure 2.7: Comparison between two similar initial conditions of (0.01, 2.0, 1.0) and (0.1, 2.0, 1.0) and the time steps of $\Delta t = 0.0073$ to show the chaotic behavior of the system. the code is at [12] and the image [13].

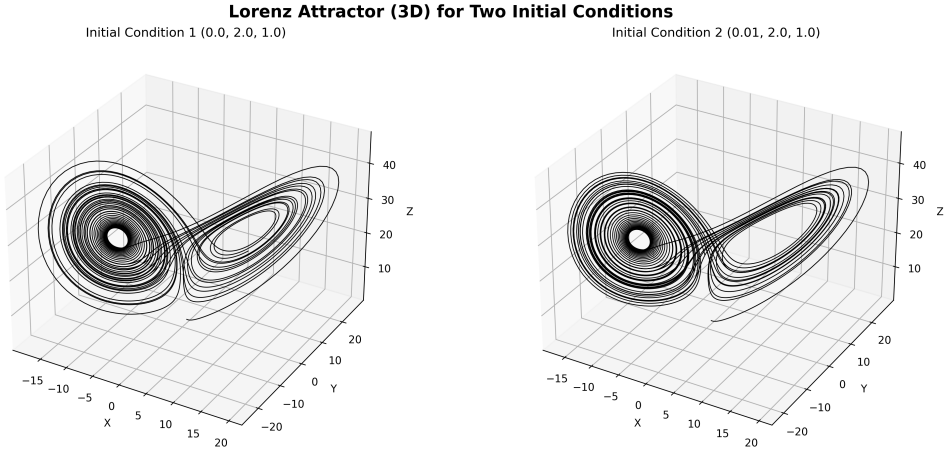

Figure 2.8: Comparison between two similar initial conditions of (0.0, 2.0, 1.0) and (0.1, 2.0, 1.0) and the time steps of $\Delta t = 0.0073$ to show the chaotic behavior of the system. the code is at [12] and the image [10].

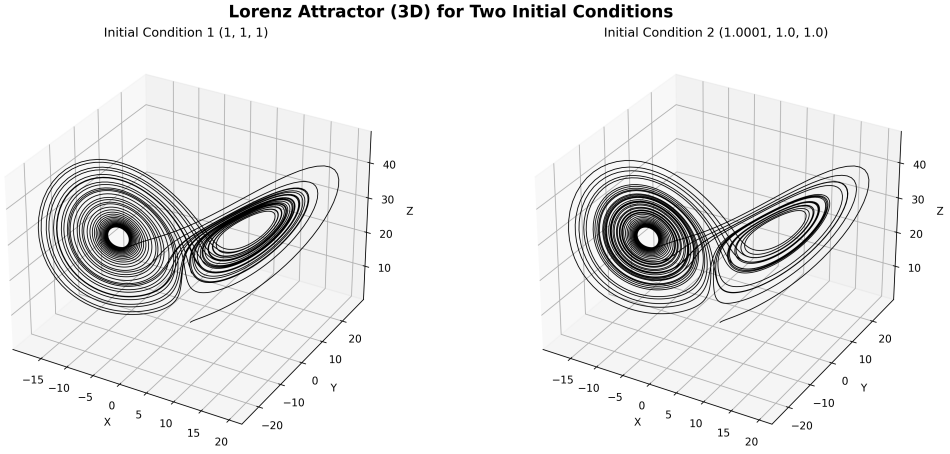

Figure 2.9: Comparison between two similar initial conditions of (1.0, 1.0, 1.0) and (1.0001, 1.0, 1.0) and the time steps of$\Delta t = 0.0073$ to show the chaotic behavior of the system. the code is at [12] and the image [11].

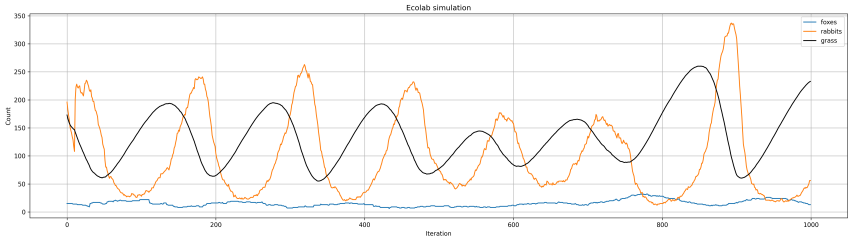

Figure 2.10: Simulation of a prey-predator system with the following initial settings: `Environment(shape=[60,60], growrate=60, maxgrass=50, startgrass=1)`, `Nrabbits = 200`, `Nfoxes = 15`. Rabbits are initialized with `speed=1`, `vision=5`, `breedfreq=10`, `breedfood=10`, and `maxage=40`. Foxes are initialized with `speed=3`, `vision=7`, `breedfreq=30`, `breedfood=20`, and `maxage=80`. See [12] for code and [11] for the source image.

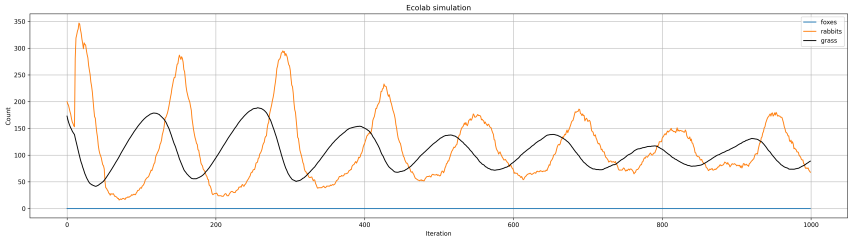

Figure 2.11: Simulation of a prey-predator system with the following initial settings: `Environment(shape=[60,60], growrate=60, maxgrass=50, startgrass=1)`, `Nrabbits = 200`, `Nfoxes = 15`. Rabbits are initialized with `speed=1`, `vision=5`, `breedfreq=10`, `breedfood=10`, and `maxage=40`. And ero foxes. See [12] for code and [11] for the source image.

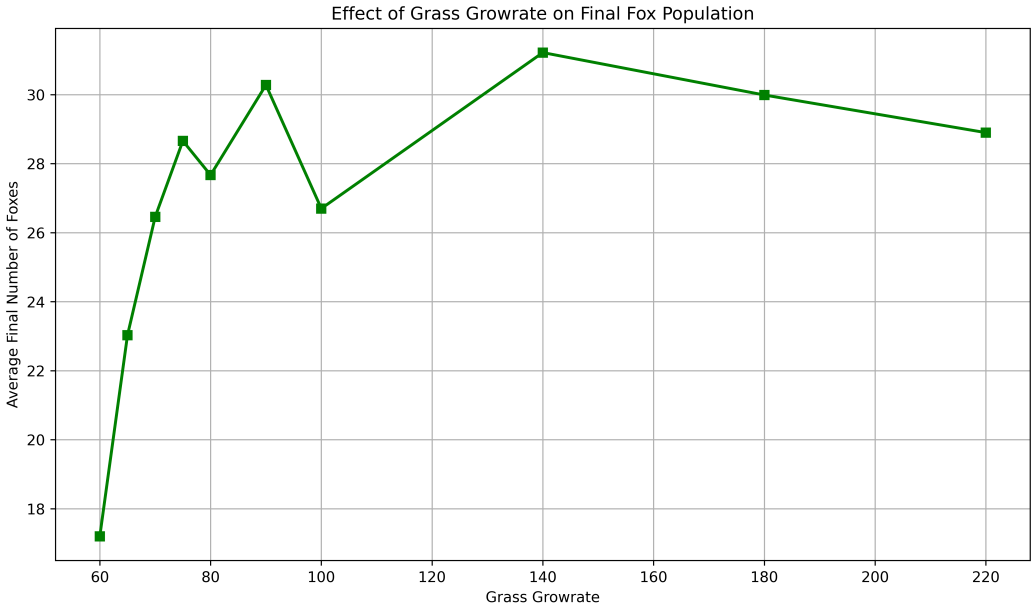

Figure 2.12: Average final foxes vs growrate, recorded across 100 independent simulation runs of the Agent-Based Model (ABM), with each run spanning up to 1000 iterations. For more information see Table 2.3.

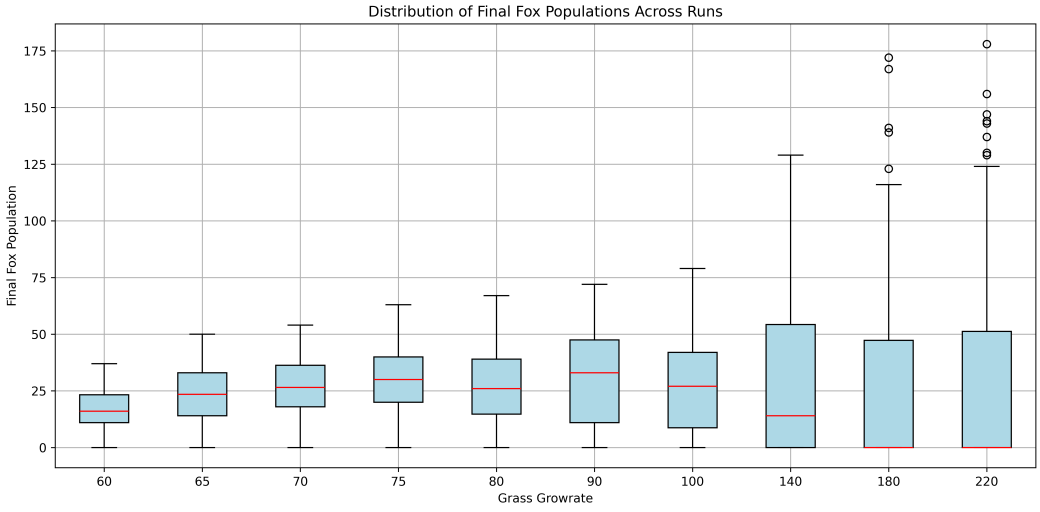

Figure 2.13: Final foxes by growrate, recorded across 100 independent simulation runs of the Agent-Based Model (ABM), with each run spanning up to 1000 iterations. For more information see Table 2.3.

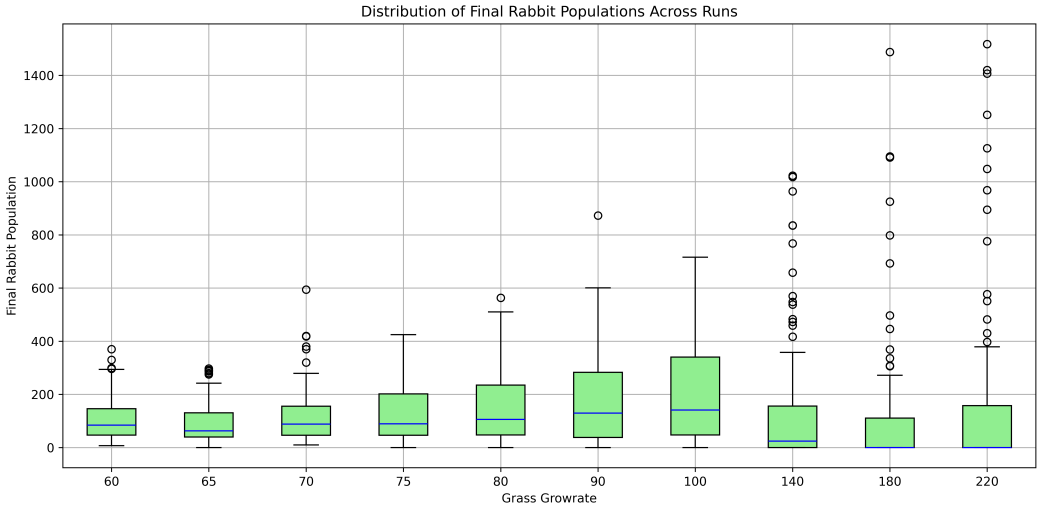

Figure 2.14: Final rabbits by growrate, recorded across 100 independent simulation runs of the Agent-Based Model (ABM), with each run spanning up to 1000 iterations. For more information see Table 2.3.

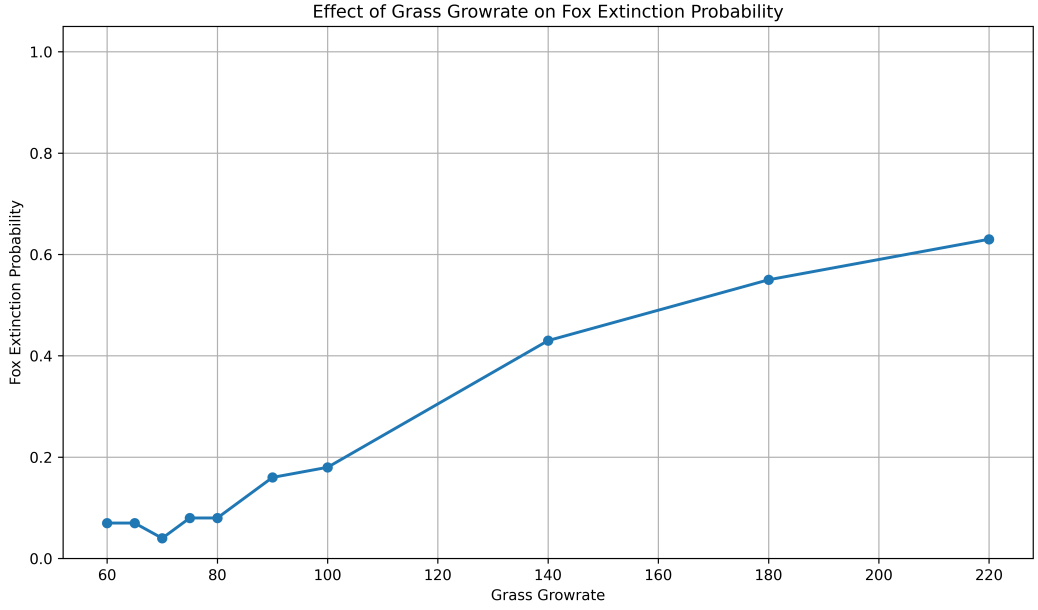

Figure 2.15: Fox extinction vs growrate, recorded across 100 independent simulation runs of the Agent-Based Model (ABM), with each run spanning up to 1000 iterations. For more information see Table 2.3.

List of Tables

Method	**Global Error**	**Pros**	**Cons**
Explicit Euler	$O(\Delta t)$	Simple; low computational cost per step; easy to implement	Poor stability for larger Δt in chaotic regimes; error accumulates linearly
RK4	$O(\Delta t^4)$	Higher accuracy and stability; better error control over fixed step sizes	More computationally expensive per step; increased complexity compared to Euler
Adaptive Methods (e.g., RK45)	Varies with adaptive step-size	Dynamically adjusts Δt based on error; accurate and efficient in varying dynamics	More complex; computational overhead for step adjustment

Table 2.1: Comparison of Integration Methods

Feature	**Agent-Based Modeling (ABM)**	**Equation-Based Modeling (EBM)**
Modeling Units	Represents individual agents, such as each rabbit or fox, capturing their unique behaviors and interactions.	Models populations as continuous variables, focusing on aggregate properties rather than individual entities.
Stochasticity	Incorporates a high degree of randomness in agents' movements, births, deaths and so on.	Typically deterministic.
Spatiality	Explicitly models space, allowing agents to move and interact within an environment (in our case, a 2D environment).	Generally lacks spatial modeling.
Emergence	Enables emergent patterns from simple interaction rules among agents.	Patterns are outcomes of governing equations; less emphasis on emergence.
Complex Interactions	Models diverse and complex behaviors at the individual level.	Simplifies interactions by averaging behaviors across the population.
Realism	More biologically realistic, but computationally expensive.	Abstract, analytically tractable, and computationally less expensive, but may lack individual-level detail.

Table 2.2: Comparison of Agent-Based Modeling (ABM) and Equation-Based Modeling (EBM)

Growrate	**Fox Extinction Probability**	**Average Final Foxes**	**Fox Std Dev**
60	0.07	17.20	9.14
65	0.07	23.03	12.35
70	0.04	26.46	13.19
75	0.08	28.66	14.62
80	0.08	27.67	16.86
90	0.16	30.28	20.86
100	0.18	26.70	20.41
140	0.43	31.22	36.62
180	0.55	29.99	44.33
220	0.63	28.90	48.93

Table 2.3: Descriptive statistics of key ecological variables recorded across 100 independent simulation runs of the Agent-Based Model (ABM), with each run spanning up to 1000 iterations. The variables include the peak, minimum, and final values for the population of rabbits and foxes, as well as the amount of grass in the environment. These metrics provide insight into the dynamics of species interactions and resource availability, helping to evaluate ecosystem sustainability and identify scenarios that may lead to population collapse or resource depletion. See Figures 2.12, 2.15, 2.13, and 2.14 for the distribution of these metrics.

Bibliography

[1] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. https://optuna.org/, 2019. Accessed: 2025-04-07.

[2] Armaos, V., Argiriou, A. A., and Kioutsioukis, I. Quantum computing and atmospheric dynamics: Exploring the lorenz system, 2024.

[3] Golbaghi, A. error$_{c}omparison_{r}k4_{s}table_{0}$.13. *https* : *//github.com/youngaryan/COM*3001$_{A}$*SSIGN/blob/main/*1.1.4/*images/error*$_{c}$*omparison*$_{r}k4_{s}table_{0}$.13. $2025 - 04 - 01$.

[4] Golbaghi, A. Lorenz attractor error 1000 time steps. https://github.com/youngaryan/COM3001$_{A}$*SSIGN/blob/main/*1.1.4/*images/error*$_{c}$*omparison*1000.*png*, 202 $2025 - 04 - 01$.

[5] Golbaghi, A. Lorenz attractor visualization. https://github.com/youngaryan/COM3001$_{A}$*SSIGN/blob/main*/ $2025 - 04 - 01$.

[6] Golbaghi, A. Lorenz attractor visualization. https://github.com/youngaryan/COM3001$_{A}$*SSIGN/blob/main*/ $2025 - 04 - 01$.

[7] Golbaghi, A. Lorenz attractor visualization. https://github.com/youngaryan/COM3001$_{A}$*SSIGN/blob/main*/ $2025 - 04 - 01$.

[8] Golbaghi, A. Lorenz attractor visualization error 3 steps. https://github.com/youngaryan/COM3001$_{A}$*SSIGN/blob/main/*1.1.4/*images/error*$_{c}$*omparison.png*, 2025.*Ac* $2025 - 04 - 01$.

[9] Golbaghi, A. optunacomparisonimage.13. https://github.com/youngaryan/COM3001$_{A}$*SSIGN/blob/main/*1 $2025 - 04 - 01$.

[10] Golbaghi, A. youngaryantwo$_{i}nitial_{c}onditions_{3}d_{2}separateicode$.13. *https* : *//github.com/youngaryan/COM*3001$_{A}$*SSIGN/blob/main/*1.1.4/*images/two*$_{i}nitial_{c}onditions_{3}d_{s}eparate_{2}$.*pn* $2025 - 04 - 01$.

[11] Golbaghi, A. youngaryantwo$_{i}nitial_{c}onditions_{3}d_{3}separateicode$.13. *https* : *//github.com/youngaryan/COM*3001$_{A}$*SSIGN/blob/main/*1.1.4/*images/two*$_{i}nitial_{c}onditions_{3}d_{s}eparate_{3}$.*pn* $2025 - 04 - 01$.

[12] Golbaghi, A. youngaryantwo$_i nitial_c onditions_3 d_s eparateicode$.13. *https* : *//github.com/youngaryan/COM*3001$_A$*SSIGN/blob/main/*1.2.1/*two$_d$if f$_i$nitial$_c$ond.py*, 2025. *Accessed* : 2025 − 04 − 01.

[13] Golbaghi, A. youngaryantwo$_i nitial_c onditions_3 d_s eparateimg$. *https* : *//github.com/youngaryan/COM*3001$_A$*SSIGN/blob/main/*1.1.4/*images/two$_i$nitial$_c$onditions$_3$d$_s$eparate.png* 2025 − 04 − 01.

[14] Kashyap, S. S., Dandekar, R. A., Dandekar, R., and Panat, S. Modeling chaotic lorenz ode system using scientific machine learning. https://arxiv.org/abs/2410.06452v1, 2024. arXiv:2410.06452v1 [cs.LG], 09 Oct 2024.

[15] Lorenz, E. N. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences 20*, 2 (Mar. 1963), 130–148.

[16] Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., and Ott, E. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science 27*, 12 (Dec. 2017).

[17] Society, A. P. Lorenz and the butterfly effect, 2003. Accessed: 2025-04-01.