MATH 340

Runge-Kutta 4th Order Methods – Lotka-Volterra Equations [EN]

Authors:

Kris Melgar Morales	cmelgarmorales@csu.fullerton.edu
Armanul Ambia	arman714@csu.fullerton.edu
Angela DeLeo	atakux707@csu.fullerton.edu

Group ♣

2023 – Spring Semester

Contents

1 Introduction

Predator-prey systems are realistic mathematical models describing the interaction between 2 species, predator and prey, within a closed environment. These systems lend well to differential equations because they are focused on the rate of change of population sizes and how they affect each other. Being able to model the trends of animal populations allows scientists to learn when populations are in danger of becoming extinct or are increasing to unsustainable rates.

This project analyzes the Lotka-Volterra model, a specific predator-prey system where there are constraints placed on each population. The prey population has access to an infinite food supply and will grow indefinitely without the influence of predators. The predator population starts at a fixed value and will continue to die without the influence of prey. The model uses a pair of equations that generate population densities between predator and prey species at each time step. These are the Lotka-Volterra Equations:

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

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

$$[\alpha, \beta, \gamma, \delta] \subseteq \mathbb{R}_+$$

$\frac{dx}{dt}$ and $\frac{dy}{dt}$ represent the "instantaneous growth rates" of the prey and predator species, x and y represent "the population density of prey" and "some predator", respectively, t represents "time", α is the "maximum prey per capita growth rate" and β is the predator's effects on the prey population's growth rate, γ is the "per capita death rate" of the predator species and δ is the prey's effect on the predator growth rate, while "all parameters are positive and real" [2].

The αx term represents exponential growth, where the prey is allowed to grow indefinitely "unless subject to predation". The βxy term is subtracted from the growth term, to represent the rate at which prey die to predators, determined by the densities of the two populations. [2]

The opposing equation defines the instantaneous growth rate of the predator population, where the δxy represent the "growth of the predator population" as the predator consumes the prey. The γy term represents the "intrinsic death rate" of the predator population. [2].

2 Implementations Details

In order to generate a graph of population densities from an initial starting point of t = 0 to an arbitrary endpoint, the Lotka-Volterra equations were solved using the Runge-Kutta Fourth Order Method defined as an algorithm in Section 4 of Chapter 5 of Numerical Analysis [1], where

$$a \leq t \leq b$$

$$w_0 = \alpha$$

is the initial condition of the function and

$$k_1 = hf(t_i, w_i)$$

$$k_2 = hf(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1)$$

$$k_3 = hf(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2)$$

$$k_4 = hf(t_{i+1}, w_i + k_3)$$

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

defines the g successive approximation of the list $[w_0, w_1, w_2, ..., w_n]$ with respect to each time step in the list $[t_0, t_1, t_2, ..., t_n]$. In our version of this implementation in Matlab, we generated the approximation with a fixed time step method, where $[t_0, t_1, t_2, ..., t_n]$ is $[0 \cdot h, 1 \cdot h, 2 \cdot h, ..., n \cdot h]$ and h is the size between each step, for a total of n+1 steps in the time interval. To solve for the Lotka-Volterra equations using the Runge-Kutta Fourth Order function, f is instead treated as a vector of anonymous functions like

$$\vec{f} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix}$$

and w_i is instead a matrix w_{ij}, where each column stores the approximations for each equation, at each time step between the rows, such as initial conditions being stored in

$$w_{0j} = \begin{bmatrix} IntitialPreyDensity & IntitialPredatorDensity \end{bmatrix}$$

3 Results

3.1 Comparison to Matlab Example

To test the accuracy of our version of the Lotka-Volterra equations using the Runge-Kutta 4th Order method, we plotted a Population Vs Time graph of our method vs the method provided by the Matlab documentation for Predator-Prey Equations [3] using these initial conditions and parameters for the equations.

$$w_{0j} = \begin{bmatrix} 20 & 20 \end{bmatrix}$$

$$\vec{f} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix}$$

$$0 \leq t \leq 15$$

$$h = 1/300, n + 1 = 4500$$

3.1.1 Matlab Plot Results

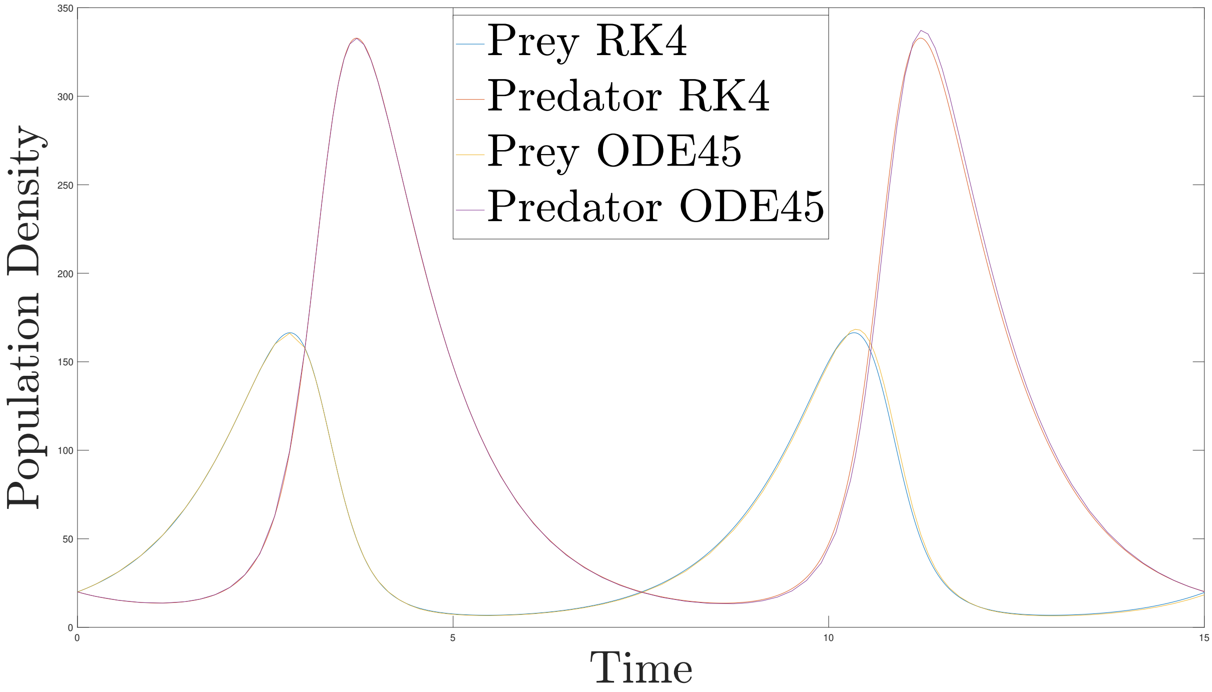

Figure 1: Predator and Prey Population Densities Over Time for Runge-Kutta 4th Order and ODE45

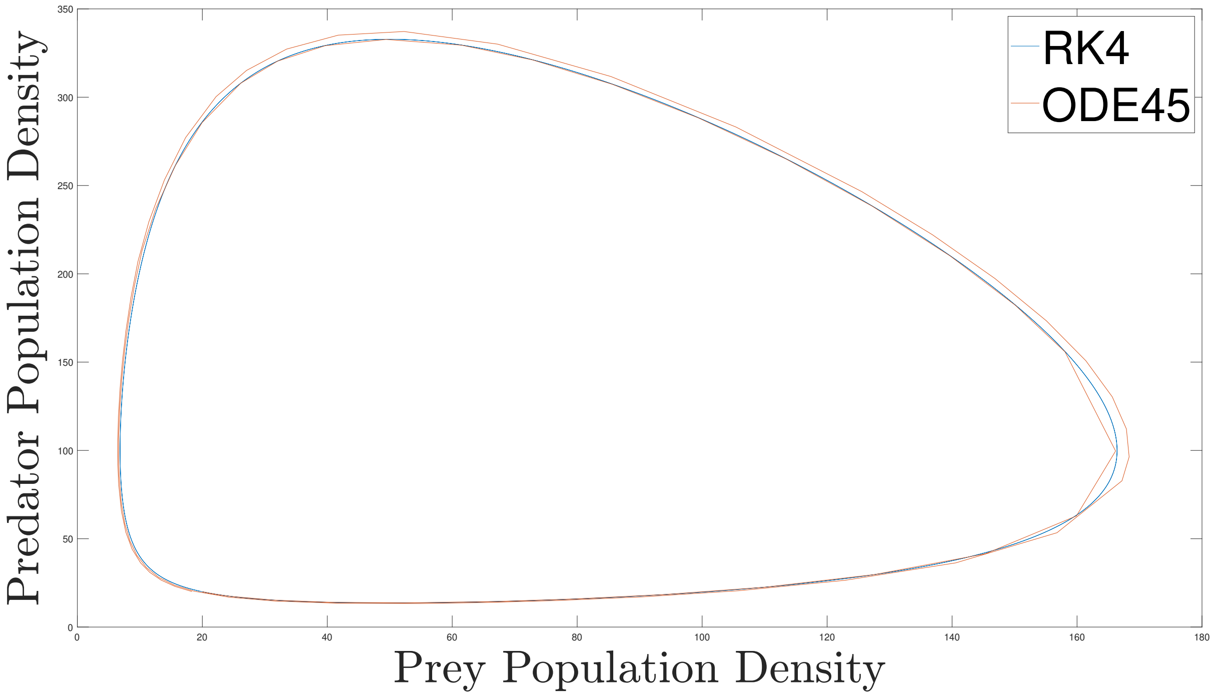

Figure 2: Predator and Prey Phase Space between Runge-Kutta 4th Order and ODE45

3.1.2 Analysis of RK4 versus ODE45

In each Figure, we can see our RK4 method is overlapping the ODE45 plots in a smoother manner, meaning our method is approximating the Lotka-Volterra Model as best as possible. Since our approximation solution is incredibly accurate to the Lotka-Volterra Model generated by ODE45, we can confirm that our method is working as expected.

4 Conclusion

In summation, the Lotka-Volterra model effectively displays how predator-prey systems can be analyzed using ODE's. The results of the experiment show that our RK4 implementation of the Lotka-Volterra model is extremely close to the approximation given by ODE45. While our results were promising, our model could be improved by testing with different initial conditions and increasing the time duration. Additionally, further work can be done to test predator-prey systems with more than 2 populations or with different constraints.

References

[1] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis 10th Editon, Chapter 5*. Cengage Learning, 2016.

[2] Wikipedia Foundation. Lotka–volterra equations. https://en.wikipedia.org/wiki/Lotka-Volterra_equations, Apr 2023.

[3] Matlab. Solve predator-prey equations. https://www.mathworks.com/help/matlab/math/numerical-integration-of-differential-equations.html.