Predator-Prey & Predatory Competition Dynamics:
A Numerical Solution Using Runge-Kutta 4th Order in C++

Andrew Goetz

December 2023

Table of Contents

Introduction

The study of predator-prey dynamics is an important area of ecology. It's analysis offers insights to the relationship(s) between species in varying environments, and how competition between two predatory species effects their population. The traditional Lotka-Volterra equations have provided a foundation for expansion. We expand on this model by considering how a single prey species reacts under two competitive predatory species, as well as by placing an environmental carrying capacity on the prey species.

There are 3 main sources that inspired this work, as well as similar studies which were used to validate/verify our results/interpretation of the model (which will be discussed later). The sources that inspired out equations are:

[1]

https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations

[2]

https://www.temjournal.com/content/61/TemJournalFebruary2017_132_136.pdf

[3]

https://www.ecologycenter.us/population-growth/the-lotkavolterra-equations.html#:~:text=The%20Lotka%2DVolterra%20model%2C%20like,itself%20consists%20of%20two%20parts.

1. Provides a fundamental understating of Lotka-Volterra and population dynamics.
2. Provides an example of a system with multiple predators.
3. Provides an overview of how to interpret results and what the system represents.

Another area for extension, not in the scope of this project was a more complex version, which introduces additional equations and variables to model the effects of climate change on the ecosystem.

This modified model seeks to illuminate the impact of environmental factors on population dynamics, emphasizing the interactions between one prey and two predators. The chosen parameters are informed by observations from ecosystems, and similar experiments/research. Many cases will draw inspiration from the relationship between rabbits and their predators like foxes and wolves, but not exclusively these relationships.

Method

The Traditional prey-predator model:

$$X` = ax - bxy$$

$$Y` = cxy - dy$$

Where a,b,c,d are constants, modeling the rate of change of prey X and predator Y.

The modified expansion includes a third predator, and an environmental carrying capacity.

(1) $\frac{dx}{dt} = ax\left(1 - \frac{x}{K}\right) - \beta_{xy}xy - \beta_{xz}xz$
(2) $\frac{dy}{dt} = -by + xy\delta_y$
(3) $\frac{dz}{dt} = -\,cz + xz\delta_z$

Where 1,2,3 model the rate of change of the prey and two predators' population respectively where x is the prey; y and z are the predators.

(1) Constant a represents natural growth rate of the prey in the absence of predators (maximum growth rate) and β_{xj} represents the predation rate (effect of predation) of j on x (often interpreted as the death rate of prey as a result of a certain predator). The term $1 - \frac{x}{K}$ represents the environmental capacity (K) which limits prey population.
(2) Constant b represents the death rate of the predator (in the absence of prey), delta Y represents the effect of the presence of prey on the predator's (Y) growth rate.
(3) Constant c represents the death rate of the predator (in the absence of prey), deltaZ represents the effect of the presence of prey on the predator's (Z) growth rate.

The system of ordinary differential equations is numerically solved using the 4th Order Runge-Kutta method. This computational approach involves iteratively updating prey and predator populations based on the calculated derivatives at each time step. The C++ implementation ensures precision in simulating the predator-prey dynamics. We use dynamic arrays to store values at each step, for each equation which are used in the plot to analyse results.

Program Tests

Verification 1

Verification is a crucial step in ensuring the accuracy and reliability of the model. In the context of the predator-prey system, it involves confirming that the implemented code represents the intended mathematical model and exhibits good behavior.

Firstly, verification of the basic Lotka-Volterra with carrying capacity from the demo code, since it implements one prey and one predator with carrying capacity (demo.cpp).

This work from Seoul National University shows how prey is supposed to react a set of environmental variables with a carrying capacity, as the population reaches that capacity:
[4] https://www.math.snu.ac.kr/~syha/appliedpde.pdf (slide 14)

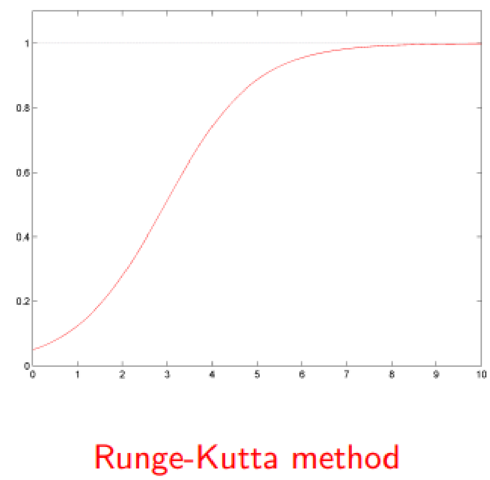

Figure 1

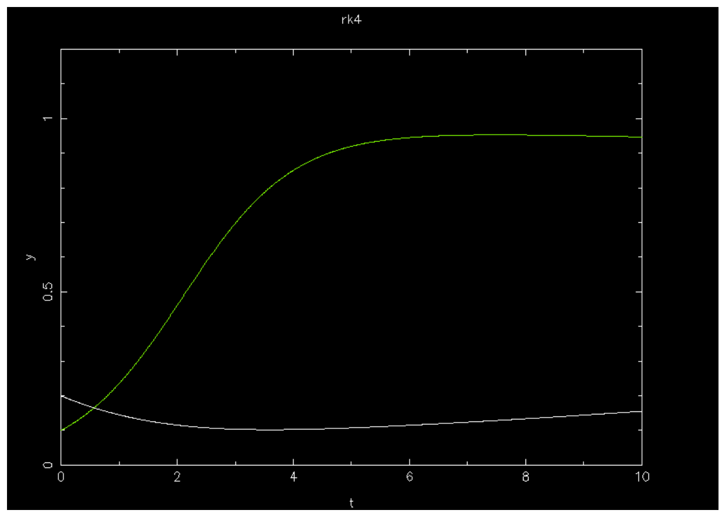

Figure 1.1 [white:predator, green:prey]

Figure 1 is what the traditional prey model should look like with a starting value < 1. On the right, Figure 1.1 is the what the un-extended version of this model looks like. It is evident that the prey follows a similar population change and reacts appropriately to both the carrying capacity (K = 1) and the increase in predator population. This is because we see the prey population retreat from the asymptote as the predator population increases.

Once again, The predator variables such as death rate, etc. were taken from the Wikipedia article on Lotka-Volterra as the traditional set of parameters for a basic model of fox-rabbit population. These are used throughout the verification step and are: growth and death rates of prey are 1.1 and 0.4 while that of predator are 0.1 and 0.4 respectively.

This test sets delta = 0.5 (effect of prey Prescence on predator growth). This model increases the delta to better represent the relationship between the species. This is done to accentuate the behaviour of the predator, due to the carrying capacity limiting growth.

Verification 2

Next, we consider the same parameters as the previous test (Figure 1.1), only with the smaller timesteps (1/10th the size). We keep time the same and increase the number of steps from 100 to 1000.

As one can see below, in Figure 1.2 the results are consistent:

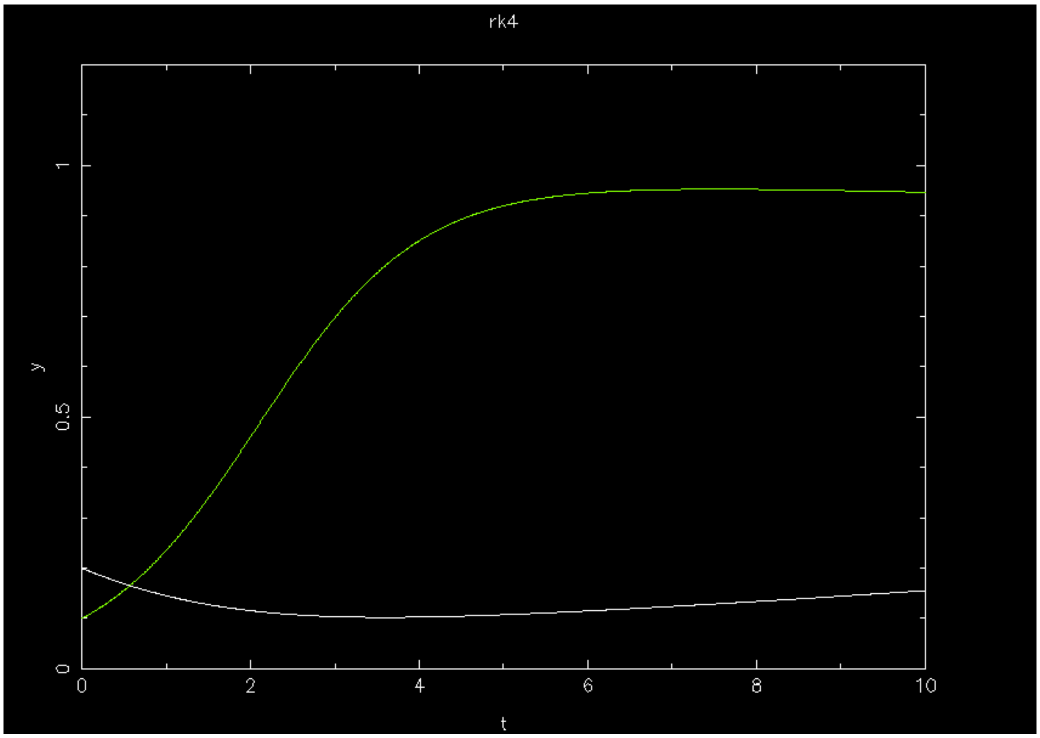

Figure 1.2

Consistency over a change to smaller time steps serves as a verification measure, ensuring the reliability of the model by confirming that incremental changes in time units still produce consistent outcomes.

Verification 3

To verify the extended model, we first consider a similar case as above, only with a new plot for the second predator. Once again we use the same parameters as above, only with two predators. The system behaves as expected with the predators behaving (exactly) the same, and the prey experiencing a slightly large decrease in population.

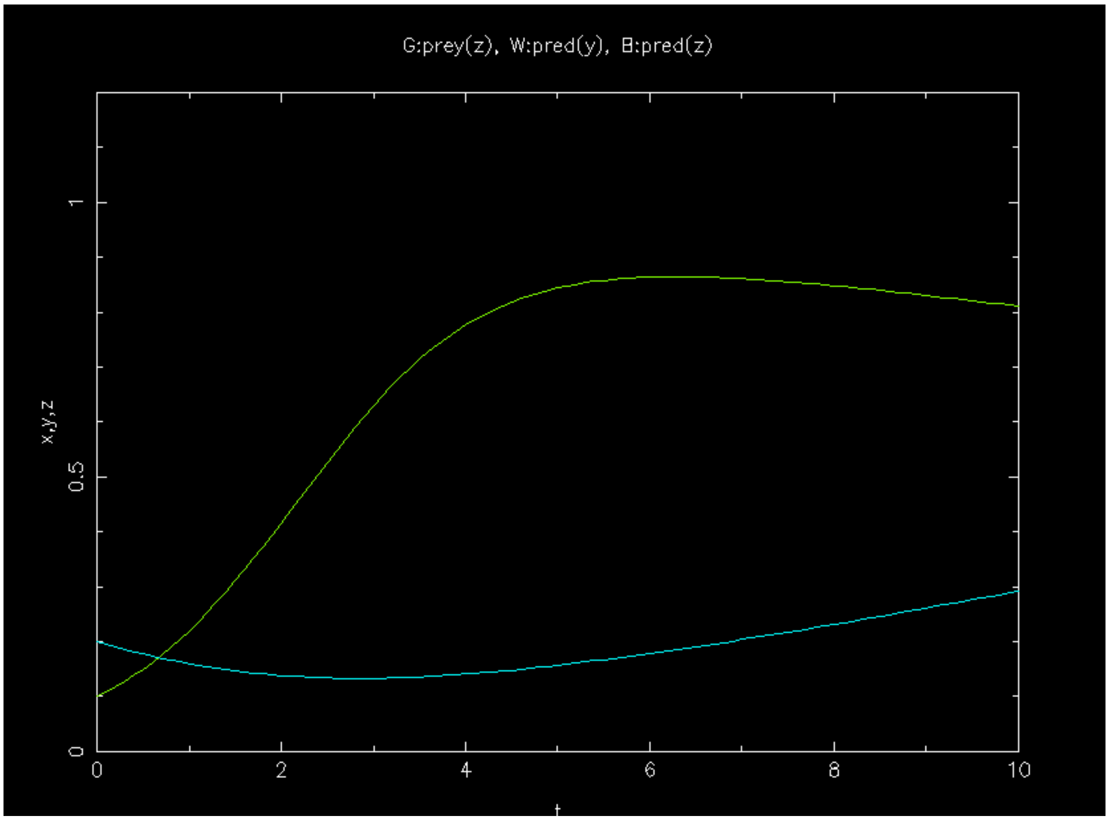

Figure 1.3 [white:predator, green:prey]

This figure of the extension behaves as expected since the two predators are modeled as identical in the environment, the white predator plot is not visible (under the blue plot) since they are identical in behaviour.

Validation 1

Next, the model will be validated, showing how the extension is a realistic model of the intended population system. The verification step already demonstrated instances of how the model (extended and basic) behaves as expected under varying conditions. This section will expand on that idea, then the results will be analyzed..

First, a test of the validity of the traditional system where delta (predator growth = 0.1). Starting populations are increased by a factor of 1.1 for both species (1.1 for prey, 2.2 for predators), and the carrying capacity is increased as well (K = 10). The experiment is run on the extended model (same as verification 3) only with the above changes to the parameters.

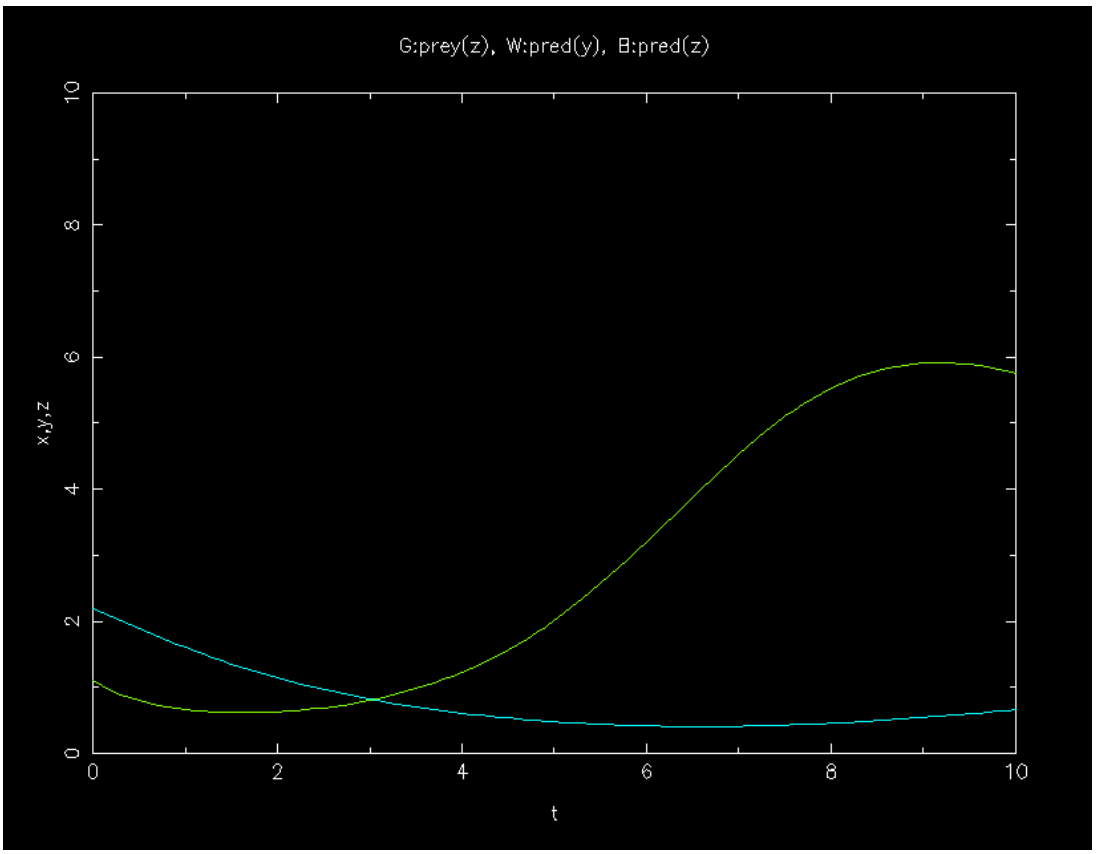

Figure 2.1

The system behaves as expected with an appropriate response from the predators in the events of both decrease and increase in prey populations. Having the carrying capacity increase so much, we can see how the system behaves in a sense closer to a traditional/basic Lotka-Volterra, while still extending the model to see how it behaves with different starting values, a much high carrying capacity and traditional delta (effect of prey on predator growth) of 0.1.

Validation 2

To show this model accurately represents a real system, compare it with an existing one. Consider what happens in the model under the same conditions as those found in a study referenced in the Wikipedia article above. This considers the demo.cpp code, without the carrying capacity term, to see how it compares to similar systems.

With 1000 steps, and end-time 100 and parameters:
Initial Prey Population (x): 0.1
Initial Predator y Population (y): 0.2
Growth Rate of Prey (a): 1.1
Death Rate of Predator y (b): 0.4
Predation Rate (Beta_xy): 0.4
Effect of Prey on Predator y (Delta_y): 0.1

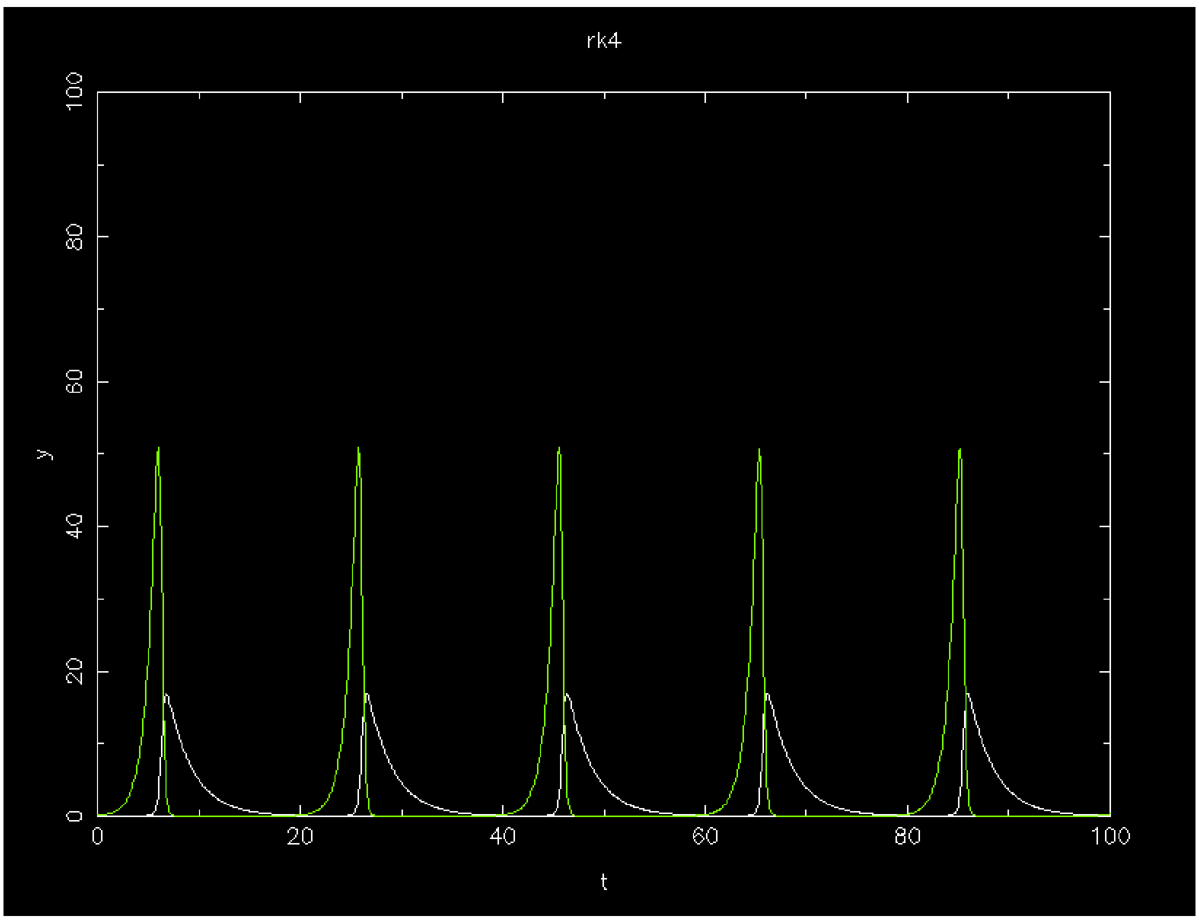

Figure 2.2

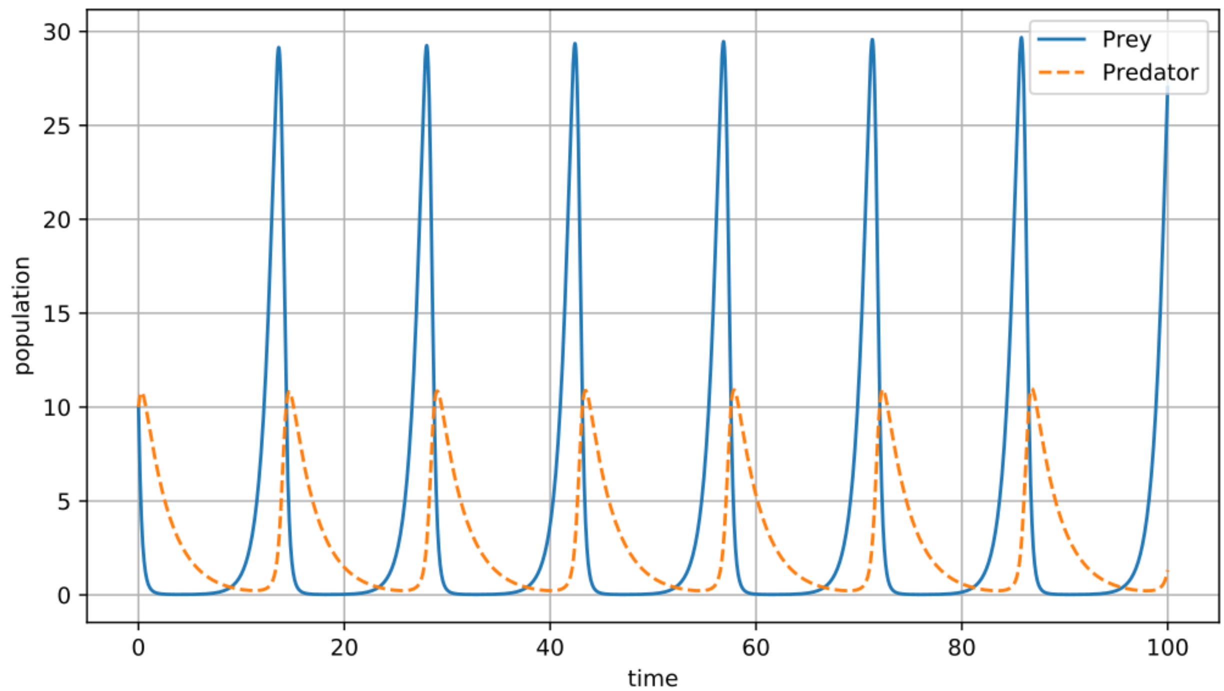

Figure 2.3

Taken from [1]:
https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations#/media/File:Lotka-Volterra_model_(1.1,_0.4,_0.4,_0.1).svg

The model behaves the same as this standard system. These parameters (growth and death rates of prey at 1.1 and 0.4; and that of predators at 0.1 and 0.4 respectively) are referenced again as the “traidional” set of parameters.

Results 1

The Results section evaluates the modified predator-prey model's performance and explores extensions. It provides an in-depth analysis of the model's behavior under various conditions, comparing outcomes to expectations and introducing novel aspects, as well as pushing beyond the limits of the model. This section critically examines the model's strengths and limitations, supported by figures.

The first test considers two predators, y and z, that commence with identical initial populations but predator z is a superior hunter, and more resilient species (*); the prey population also starts at the same level. The chosen parameters for the model are as follows:

Initial Prey Population (x): 2
Initial Predator y Population (y): 2
Initial Predator z Population (z): 2
Growth Rate of Prey (a): 1.1
Death Rate of Predator y (b): 0.4
Death Rate of Predator z (c): 0.3 * Stronger species
Predation Rate (Beta_xy): 0.3
Predation Rate (Beta_xz): 0.8 * Predator z is a more efficient hunter
Environmental Capacity (K): 1
Effect of Prey on Predator y (Delta_y): 0.5
Effect of Prey on Predator z (Delta_z): 0.6 *

Used in pred_prey.cpp (extended model).

Despite the equal starting conditions, the predator z is assigned a superior death rate compared to predator y. This differential parameterization aims to explore how subtle variations in the system's parameters can lead to distinct ecological outcomes over time. The delta values for both predators are increased to accentuate the results of the model.

Through this case, we can validate how the model represents scenarios where predators exhibit varied population due to slight differences in their interaction with the environment and predatory ability. This also shows what conditions might lead to eventual extinction (of Y).

Figure 3.1 provides insights; the effects of a predator z (blue) having such efficient and advanced hunting sees its rapid incline in population leading to a decrease in prey population over time as well.

Another valuable insight from this figure, is that predator y (white) is slightly decreasing in population despite having parameters that, in a system without predator z, would be favourable. Consider figure 3.2, that shows what would happen in the exact same scenario with the presence of predator z entirely.

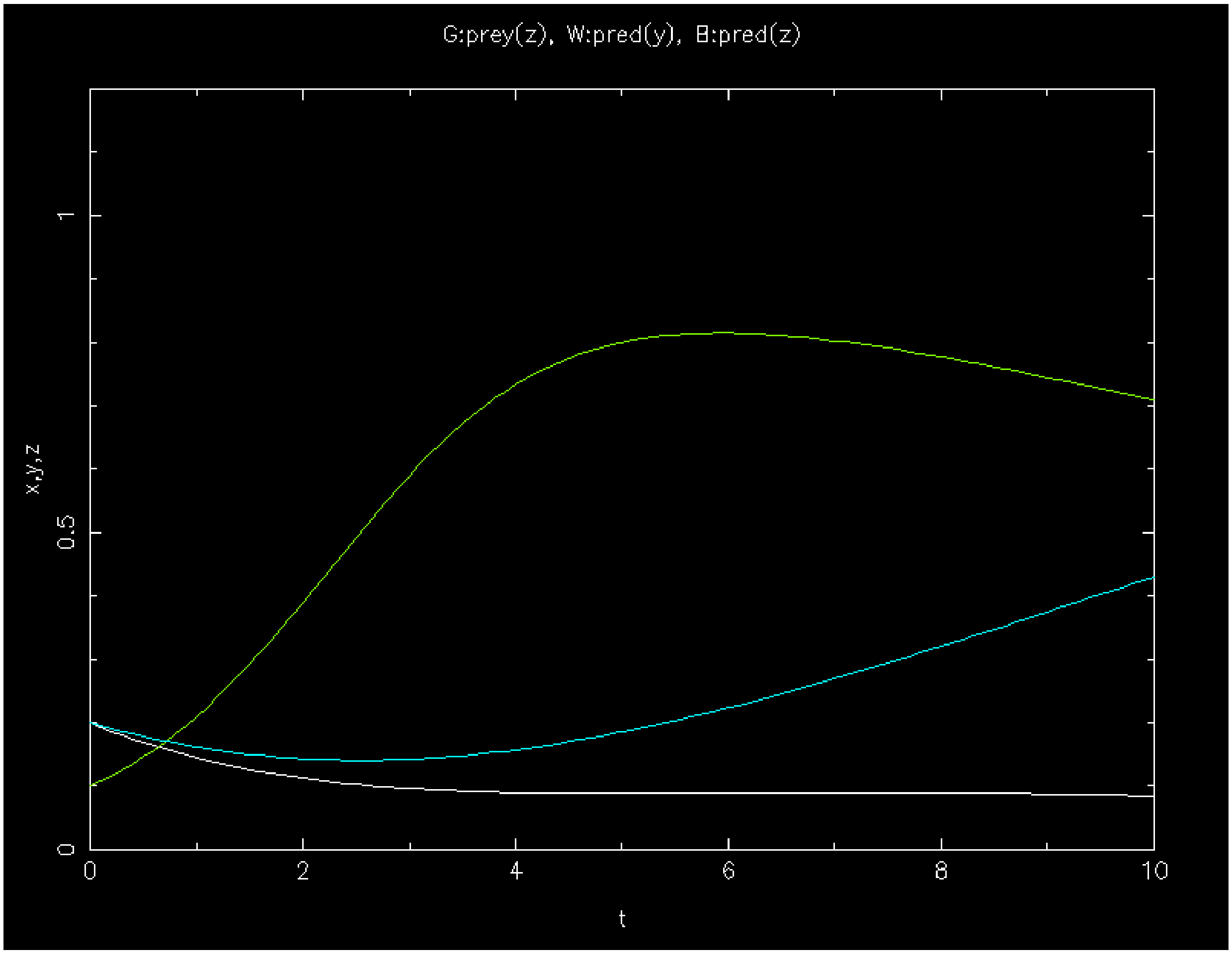

Figure 3.1

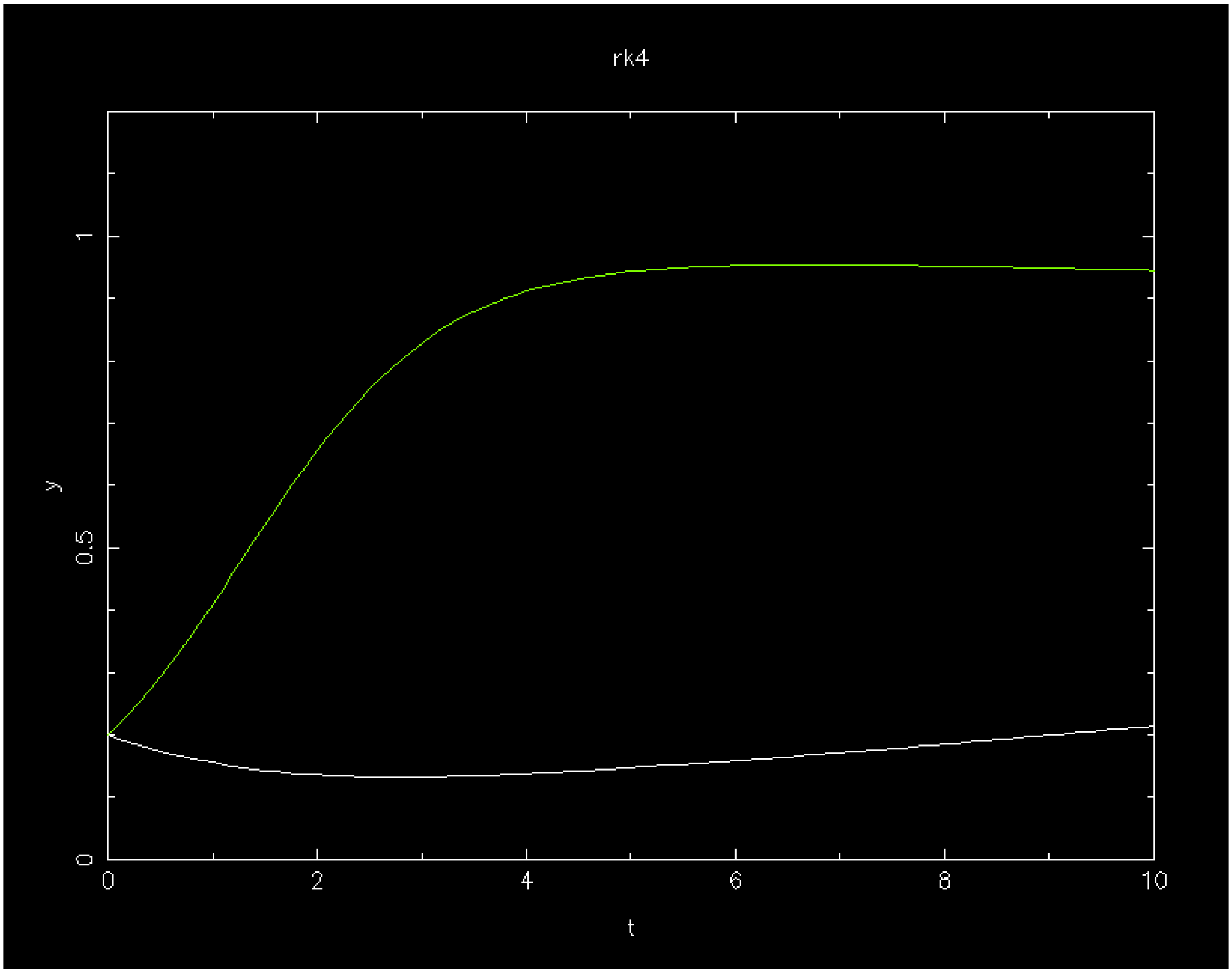

Figure 3.2 [Shows just one predator, taken from demo.cpp]

Results 2

The next results analysis involves a system with an unusually high death rate in one of the predators, specifically in predator y, provides intriguing insights into the dynamics of the modified predator-prey model. The elevated death rate for predator y (b = 0.8) introduces a significant imbalance between the two predators, creating a scenario where predator z faces less competition and a lower death rate (c = 0.2). The goal here is to show one predator outc-competing another.

The parameters for the system are as follows:

Unusually high death rate (in y):
Initial Prey Population (x): 0.1
Initial Predator y Population (y): 0.2
Initial Predator z Population (z): 0.2
Growth Rate of Prey (a): 1.1
Death Rate of Predator y (b): 0.8 **Unusually high death rate for predator y **
Death Rate of Predator z (c): 0.2 ** lower death rate for z **
Predation Rate (Beta_xy): 0.3
Predation Rate (Beta_xz): 0.3
Environmental Capacity (K): 1
Effect of Prey on Predator y (Delta_y): 0.4
Effect of Prey on Predator z (Delta_z): 0.4

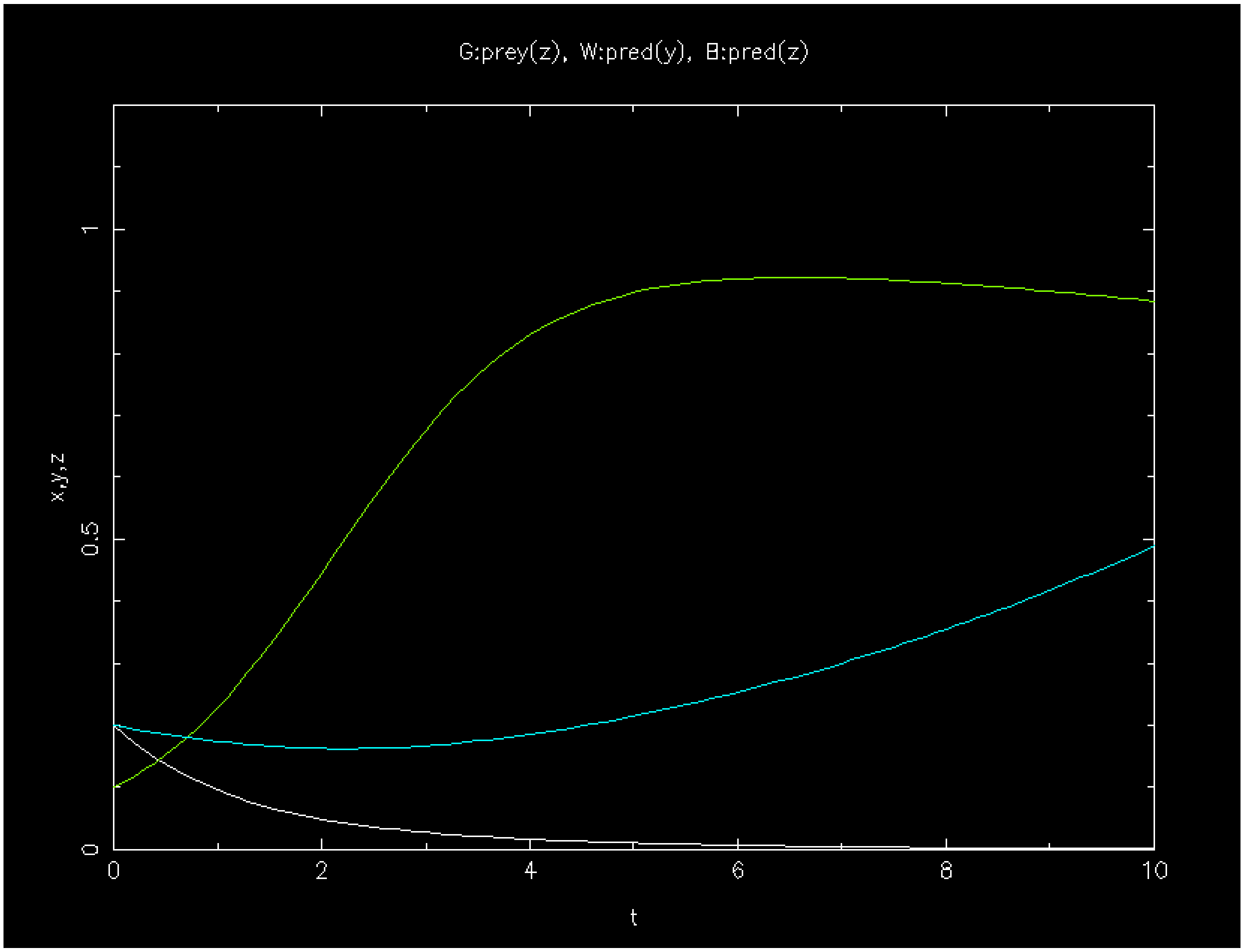

Figure 3.3

The results show that, despite the shared parameters between predators for predation rates (Beta_xy and Beta_xz) and environmental capacity (K), predator y struggles to maintain its population due to the elevated death rate. This leads to a notable increase in the population of predator z, which benefits from a less competitive environment. The prey population experiences fluctuations, influenced by the imbalanced predator dynamics.

The figures generated from this test accurate models a hypothetical system and visually illustrates the impact of an unusually high death rate on the predator-prey interactions, providing valuable insights into the population's behavior under specific conditions.

The imbalance in death rates between predators y and z, as seen in Figure 3.3, illuminates how subtle variations in predator parameters can lead to distinct ecological outcomes. This emphasizes the model's ability to capture nuanced changes in dynamics within a system, having one predator outcompete another.

Results 3

Lastly, consider a set of parameters designed to push the model. This is to simulate a scenario that the current model may struggle to accurately capture.

The following set of parameters introduces a prey that has a much-increased growth rate, while predators maintain stable parameters:

Initial Prey Population (x): 0.1
Initial Predator y Population (y): 0.2
Initial Predator z Population (z): 0.2
Growth Rate of Prey (a): 1.9 ** increased prey growth rate
Death Rate of Predator y (b): 0.4
Death Rate of Predator z (c): 0.4
Predation Rate (Beta_xy): 0.3 * slight difference in predation rates
Predation Rate (Beta_xz): 0.4
Environmental Capacity (K): 1
Effect of Prey on Predator y (Delta_y): 0.15
Effect of Prey on Predator z (Delta_z): 0.09

The introduced complexities, such as a higher prey growth rate and lower delta values, has revealed challenges for the model in accurately. It is representing a scenario where predator populations struggle despite an abundance of prey, and standard parameters. This underscores the model's limitations, particularly with environmental capacity and specific predator-prey dynamics.

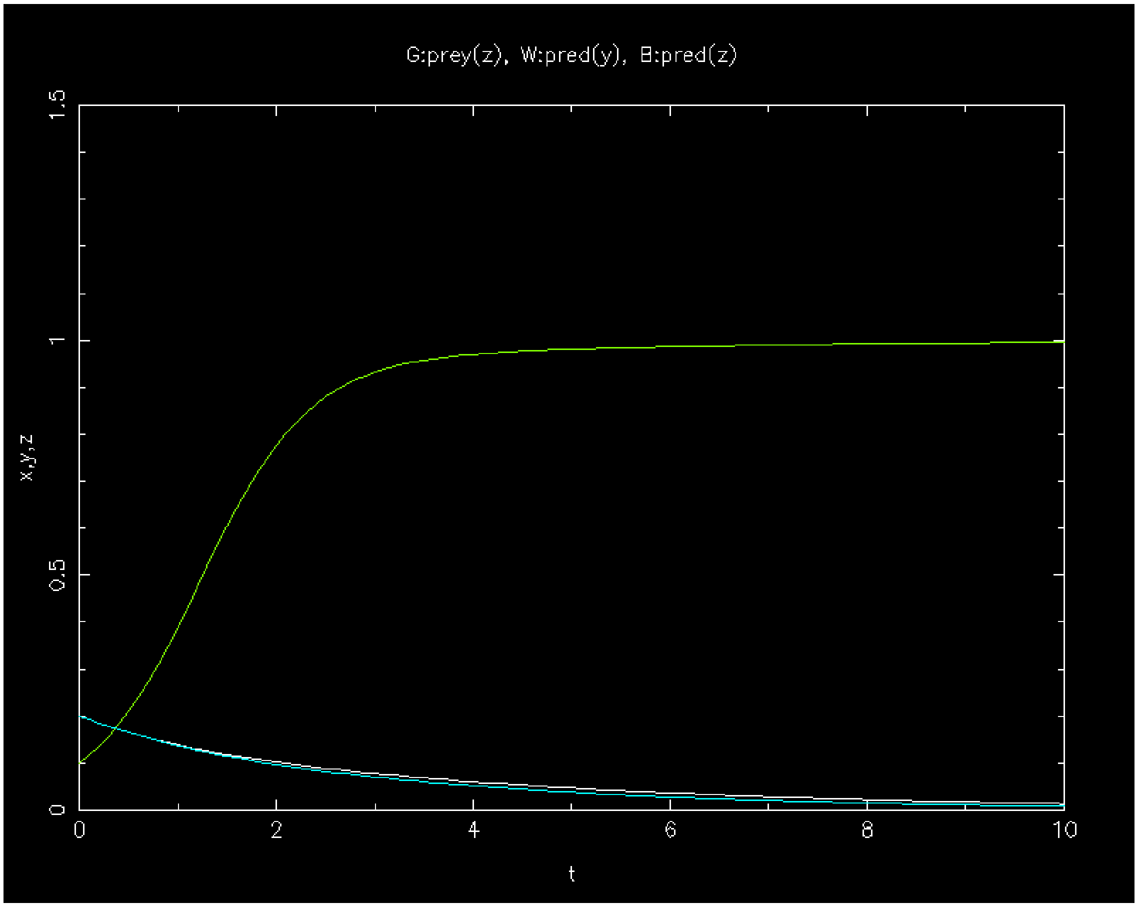

Figure 2.4

The extended model reveals that the prey quickly reaches its carrying capacity, while both predator species experience a gradual decline in population.

The chosen parameter set introduces complexities that may be challenging for the model, shedding light on its limitations. Despite an abundance of prey, the predators struggle, likely influenced by the carrying capacity restricting prey populations (from going as high as necessary). The lower delta values in this test challenge the model's accuracy. The values provided for delta are not extreme, in fact they are common yet, the system cannot overcome the implementation of the carrying capacity.

The combination of high prey growth rate, elevated environmental capacity, and specific predator-prey interaction rates may yield unrealistic outcomes. The expected results might be that predator populations are impacted by prey growth rate bot the following figure (2.5) suggest otherwise. Taken from the same model, only with prey growth (a) changed from 1.9 to 1.1, it is evident that predators see little difference from this change.

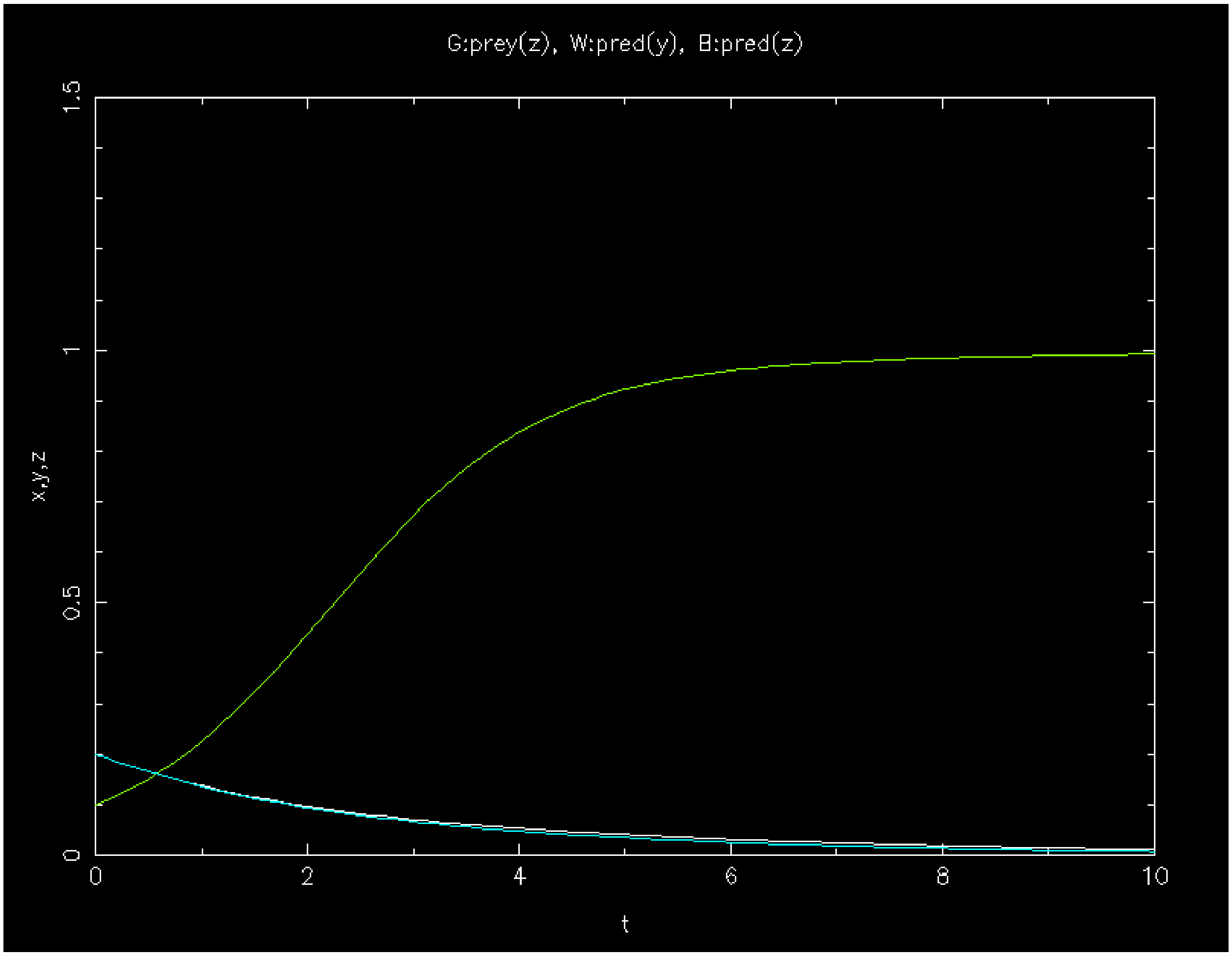

Figure 2.5

Lastly, it is valuable to note that Predator y (white) displays greater competitiveness, making the smaller difference in delta (difference of 0.6) more impactful than a 0.1 increase in beta for predator z (blue).

Conclusions

The exploration of our modified Lotka-Volterra model unveils insights into predator-prey dynamics, with two predators and a carrying capacity.

Figure 3.1 highlights the intriguing scenario where a superior predator (predator z) encounters competition from another predator (predator y), resulting in a decline in the inferior predator, and perhaps eventually, extinction. With it's advanced hunting abilities and resilience in the environment, predator z's rapid increase influences the complex dynamics within the system. Figure 3.2, depicting the scenario with only predator y, underscores the importance of understanding competitive interactions among predators, as predator y experiences a more favorable population trend in the absence of competition.

The examination of the extended model offers another valuable insight into the delicate balance of predator-prey relationships. Figure 3.3 explores a scenario where predator y faces an unusually high death rate, resulting in an imbalanced dynamic with predator z. This imbalance leads to a substantial impact on the prey population, highlighting the sensitivity of the system to variations in predator parameters.

This study suggests potential avenues for alternative approaches, such as refining the model to incorporate spatial considerations or additional environmental variables. Future work could investigate the impact of climate change on predator-prey dynamics.

In conclusion, the modified Lotka-Volterra model serves as a versatile framework for studying predator-prey dynamics, emphasizing the significance of competition dynamics among predators, with environmental carrying capacity. The insights gained lay the groundwork for further exploration into the complexities of competition and predation.

Implementation

Variables:
a: Growth rate of prey.
b: Death rate of the first predator.
c: Death rate of the second predator.
K: Environmental carrying capacity for prey.
beta_xy: Predation rate of the first predator on the prey.
beta_xz: Predation rate of the second predator on the prey.
delta_y: Effect of prey presence on the growth rate of the first predator.
delta_z: Effect of prey presence on the growth rate of the second predator.
x: Prey population.
y: Population of the first predator.
z: Population of the second predator.

The C++ code implements a modified Lotka-Volterra model to simulate predator-prey dynamics. The model “pred-prey.cpp” extends the code from “demo.cpp” to incorporate another predator. The main function invokes a 4th Order Runge-Kutta method (rk4) to numerically solve the system.

The simulation runs for a specified number of steps, and the results are plotted using the PGPLOT library, distinguishing prey and predator populations over time.

Code on next page.

pred_prey.cpp (extended model)

```
#include <iostream>
#include <cmath>
#include <math.h>
#include "cpgplot.h"
#include <cstdlib>

// declare functions
double rk4(int steps);
double f_x( double x, double y, double z, double a, double K, double beta_xy,
double beta_xz);
double f_y( double x, double y, double b, double delta_y);
double f_z( double x, double z, double c, double delta_z);

// main function that calls solution
int main(){
rk4(100);

return 0;
}


// Define the differential equations
double f_x( double x, double y, double z, double a, double K, double beta_xy,
double beta_xz) {
    return a * x * (1 - (x / K)) - (beta_xy * x * y) - (beta_xz * x * z) ;
}

double f_y( double x, double y, double b, double delta_y) {
    return -b * y + (x * y * delta_y);
}

double f_z( double x, double z, double c, double delta_z) {
    return -c * z + (x * z * delta_z);
}



double rk4(int steps){
// parameters
    double a = 1.1;     // growth rate of prey
    double b = 0.4;     // death rate of predator (per capita) Y
    double c = 0.4;    // " Z
    double K = 1;     // environmental capacity
    double beta_xy = 0.3;  // predation rate
    double beta_xz = 0.4;  // (often intepretted as prey death rates, as a
result of a particular predator)

    double delta_z = 0.09; // effect of prey prescence on predator growth rate/
(aka predator growth rate)
```

```
    double delta_y = 0.15;

// user input delta (for testing
    double delta;
    //std::cout << "delta: \n";
    //std::cin >> delta;
        //delta_y = delta; delta_z = delta;

//initial values
    double x = 0.1; double y = 0.2; double z = 0.2;

// time
    double tstart = 0.0;
    double tend = 10.0;
    double d = (tend - tstart)/steps;
    double t = tstart;

        // arrays for plot
    float* tp = (float *)calloc(steps +1, sizeof(float));
    float* yp = (float *)calloc(steps +1, sizeof(float));
    float* xp = (float *)calloc(steps +1, sizeof(float));
        float* zp = (float *)calloc(steps +1, sizeof(float));

        int n = steps;
    for (int i = 0; i < n; i++){
        tp[i] = t;
        yp[i] = y;
        xp[i] = x;
                zp[i] = z;

// iterative rk4
        double k1x = d * f_x( x, y, z, a, K, beta_xy, beta_xz);
        double k1y = d * f_y( x, y, b, delta_y);
        double k1z = d * f_z( x, z, c, delta_z);

        double k2x = d * f_x( x + 0.5 * k1x, y + 0.5 * k1y, z + 0.5 * k1z, a,
K, beta_xy, beta_xz);
        double k2y = d * f_y( x + 0.5 * k1x, y + 0.5 * k1y, b, delta_y);
        double k2z = d * f_z( x + 0.5 * k1x, z + 0.5 * k1z, c, delta_z);

        double k3x = d * f_x( x + 0.5 * k2x, y + 0.5 * k2y, z + 0.5 * k2z, a,
K, beta_xy, beta_xz);
        double k3y = d * f_y( x + 0.5 * k2x, y + 0.5 * k2y, b, delta_y);
        double k3z = d * f_z( x + 0.5 * k2x, z + 0.5 * k2z, c, delta_z);

        double k4x = d * f_x( x + k3x, y + k3y, z + k3z, a, K, beta_xy,
beta_xz);
        double k4y = d * f_y( x + k3x, y + k3y, b, delta_y);
        double k4z = d * f_z( x + k3x, z + k3z, c, delta_z);
```

```
        t = t + d;
        x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
        y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
                z = z + (k1z + 2 * k2z + 2 * k3z + k4z) / 6;
        }
        tp[n] = t; yp[n] = y; xp[n] = x;  zp[n] = z;

//plot
if (!cpgopen("/XWINDOW")) return 1;

cpgenv(0., tend, 0., 1.5, 0, 1);
cpglab("t", "x,y,z", "G:prey(z), W:pred(y), B:pred(z)");

cpgsci(1); // White: predator
cpgline(n+1, tp, yp);


cpgsci(9); // Green: prey
cpgline(n+1, tp, xp);

cpgsci(5); // blue: prey2
cpgline(n+1, tp, zp);

cpgclos();


return 0.0;
}
```

Continued on next page

Demo.cpp (One predator, One prey)

```
#include <iostream>
#include <cmath>
#include <math.h>
#include "cpgplot.h"
#include <cstdlib>


double rk4(int steps);
double f_x( double x, double y, double a, double K, double beta);
double f_y( double x, double y, double b, double delta);

int main(){
rk4(100); // number of steps

return 0;
}


// Define the differential equations
double f_x( double x, double y, double a, double K, double beta) {
    return a * x * (1 - (x / K)) - (beta * x * y);}


// Alternate f_x without carrying capacity, used in verification
//double f_x( double x, double y, double a, double K, double beta) {
  //  return a * x *1/* (1 - (x / K))*/ - (beta * x * y); }

double f_y( double x, double y, double b, double delta) {
    return -b * y + (x * y * delta);
}




double rk4(int steps){

    double a = 1.1;     // growth rate of prey
    double b = 0.4;     // death rate of predator (per capita)
    double K = 1;     // environmental capacity
    double beta = 0.4;//0.4;  // predation rate
    double delta = 0.5; // effect of prey prescence on pred/ predator
effeciency
                     // This variable may be the focus of my experiments
                     // And demo.
    // Notes from DEMO:
        // How does changing predator effeciency (effect of prey presceence on
growth rate) change the curve?
        // Consider these values: 0.4
        // 0.65
```

```
//user input
 //std::cout << "delta: \n";
   // std::cin >> delta;

    double x = 0.1; double y = 0.2;

    double tstart = 0.0;
    double tend = 10.0;
    double d = (tend - tstart)/steps;
    double t = tstart;

    float* tp = (float *)calloc(steps +1, sizeof(float));
    float* yp = (float *)calloc(steps +1, sizeof(float));
    float* xp = (float *)calloc(steps +1, sizeof(float));

        int n = steps;
    for (int i = 0; i < n; i++){
        tp[i] = t;
        yp[i] = y;
        xp[i] = x;

        double k1x = d * f_x( x, y, a, K, beta);
        double k1y = d * f_y( x, y, b, delta);

        double k2x = d * f_x( x + 0.5 * k1x, y + 0.5 * k1y, a, K, beta);
        double k2y = d * f_y( x + 0.5 * k1x, y + 0.5 * k1y, b, delta);

        double k3x = d * f_x( x + 0.5 * k2x, y + 0.5 * k2y, a, K, beta);
        double k3y = d * f_y( x + 0.5 * k2x, y + 0.5 * k2y, b, delta);

        double k4x = d * f_x( x + k3x, y + k3y, a, K, beta);
        double k4y = d * f_y( x + k3x, y + k3y, b, delta);

        t = t + d;
        x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
        y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;

        }
        tp[n] = t; yp[n] = y; xp[n] = x;


if (!cpgopen("/XWINDOW")) return 1;

cpgenv(0., tend, 0., 1.2, 0, 1);

cpglab("t", "y", "rk4");

cpgsci(1); // White: predator
cpgline(n+1, tp, yp);
```

```
cpgsci(9); // Green: prey
cpgline(n+1, tp, xp);

cpgclos();


return 0.0;
}
```

Works Cited

[1]

Wikimedia Foundation. (2023, November 8). *Lotka–Volterra equations*. Wikipedia. https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations

[2]

Lotka-Volterra model with two predators and their prey - TEM Journal. (n.d.). https://www.temjournal.com/content/61/TemJournalFebruary2017_132_136.pdf

[3]

Doty, L. (2023, July 6). *The Lotka Volterra equations* - *population growth*. Ecology Center. https://www.ecologycenter.us/population-growth/the-lotkavolterra-equations.html#:~:text=The%20Lotka%2DVolterra%20model%2C%20like,itself%20consists%20of%20two%20parts.

[4]

Seung Yeal ha - seoul national university. (n.d.-b). Logistic Map, Euler & Runge-Kutta Method and Lotka-Volterra Equations. https://www.math.snu.ac.kr/~syha/appliedpde.pdf