“The impact of harvesting on the Lotka-Volterra model”

by

Candidate - 861

Introduction

The world we humans live in is a strange one. We humans can't even start to phantom all the different parts that make up our existence. The way most people view life is through approximations and generalizations. In order to better understand our complicated reality, we create models and try to break it down. This is why our school system has different grades; we simplify our explanation when talking to a non-expert and, in general, remove the non-necessary components of anything we are trying to understand. This can be seen in our daily life. The news, our books, and in the movies. They create a representation of the world and make it familiar for us to understand the context but strip away anything that might not help the story or our understanding.

In the movie "Lion King," the opening scene is of the animals hearing the call for a new lion being born. All the animals come together, and we see them rejoice at the fact that they have got a new prince and future leader. This is not exactly how the animal kingdom works. You can ask David Attenborough. However, when trying to explain to kids how the jungle works, they might have to simplify. This is a cartoon movie for children, so the scientific validity of their representation will not be under any scrutiny. Yet they have created a model in which their story can be told. This is something that happens in science as well. A question is posed, and we want to understand more and get a better sense of what is going on. We, therefore, construct a model where we can remove the distractions and focus on the essential parts.

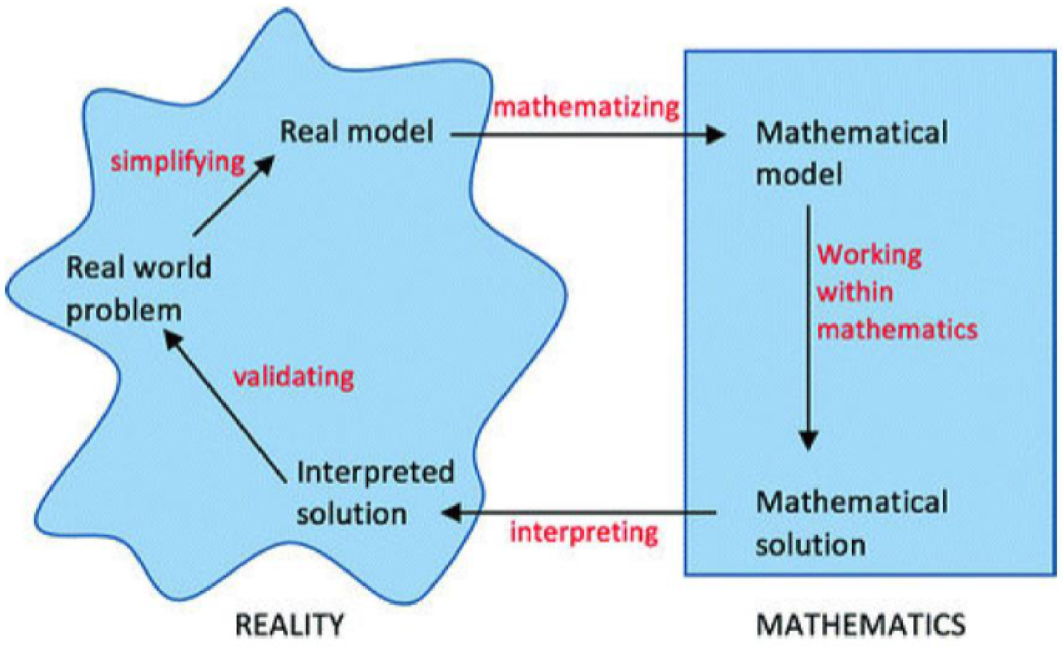
In this report, we will be looking at such a model. We will be discussing how we got this model, what a model is, and how do we use it to get a better understanding of our world. This report will take a closer look at the famous Lotka-Volterra model. It's a model that looks at the relationship between a prey and its predator. In a roundabout way, I would like for us to dive a little deeper into the opening song from Lion King, "Circle of Life."

Background and model description

In this chapter, we will be looking at some of the concepts that we will use to understand this report. First, we will look at what a mathematical model is and what our mathematical model is actually representing.

Mathematical Model

Most of our understanding of this world comes from creating models. We develop models to make complex systems easier to understand. We remove nonessential functions and try to make a real enough model to explain the underlying mechanics, dynamics, and increases of the world around us. For this, we follow a specific process. This process can be seen in the diagram below [15]:



This diagram shows us the circle of life that a mathematical model has. It usually starts with us having a real-world problem that we are interested in solving. This leads us to boil it down to its core parts and create an accurate world model. Next, we formulate this simplified version of the problem as a mathematical model. This means that we usually make an equation that describes the model. Now that we have mathematized the model, we can start working within the realm of mathematics and try to see simulate, extrapolate and better understand the model. When we have a conclusion mathematically, we then turn back to reality and see if our mathematical solution aligns with the real world. This process is usually an iterative process that takes many rounds before we can be satisfied with the mathematical model.

Prey-predator model

The model chosen for this project is a variation of the famous Lotka-Volterra model. The model in its proper form describes the relationship between prey and predators and can be expressed as such:

$$x' = x(a - by)$$

$$y' = y(cx - d)$$

If we look at the original model, we can see the relationship between prey and predator is closely related. If we look at the individual terms in this equation, we will see how they are linked.

Listed below is the meaning of each variable:

- $x = prey$
- $y = predator$
- $a = birthrate$
- $b = death\,rate$
- $c = birthrate$
- $d = death\,rate$

The equations are closely linked. We can see that the number of predators impacts the number of prey, and the number of predators is affected by the number of prey, which corresponds to our understanding of how nature works. As previously stated, models usually remove a lot of factors to simplify. This model is based on the condition that the predators don't have any other source of food or any other real-world scenarios take place. The only impact on these two parties is each other. The more prey that is born, the more predators will be born, which in turn will reduce the number of prey born and again negatively impact the predators, giving rise to the prey population. It's a balancing act that legitimizes the term "Circle of Life."

The equation presented above is classified as an ordinary differential equation, better known by its acronym, ODE. It is composed of two first-order nonlinear equations. Where the x' represents the change in prey over time and y' corresponds to the shift in the number of predators over time. These ODEs are useful when trying to describe a system. Because we can then see how this system changes over time. In order to see this, we use numerical methods. This means solving it with a computer. We will get into that a bit later. One can also analyze this by hand, which we will also see in the next chapter. But the main thing to understand is that this model is a dynamic system that changes over time, depending on the different values of the constants. We choose different values for a,b,c and d, and we will see that the behavior of our model will also change with it. There are also variations of this model. The variation we will look at for this project has an added constant, harvesting.

$$x' = x(a - h - by)$$

$$y' = y(cx - (d + h))$$

How does harvesting impact our model? This is what we will try to answer in this report. In the next chapters, we have done a numerical analysis of the model, trying to see how it behaves and how our harvesting rates change that behavior. We have also used numerical methods and see how our system evolves over time.

Numerical Analysis

When working with models, we usually have a dynamic system that changes over time. We can simulate this change over time with numerical methods. First, however, the numerical analysis gives us a general idea of how this system works.

The Lotka-Volterra model is a two-dimensional system. In such a system, we can plot the values of the system as a vector field. The x-axis shows the magnitude of x, and the y-axis shows the magnitude of y. Any point on the vector field (x,y) can be an initial condition for our system. A vector denotes the direction our system will move. What can be seen is that if we plot different vectors in this phase plot, a trajectory will emerge, showing us how the system is moving. This is not the change in time. However, it will show us what the system is converging into. The system is looking for its equilibrium.

For the prey-predator model, two points are inherently different than the rest. These are the balanced or steady points of the population, and 0. When one of these values is reached, the population does not change—giving us a steady state. These steady-state values can be represented on a phase plot. Beyond just finding these steady states, we can now see if they are stable. A stable, steady point is where any change to the system will ultimately revert to this state. For example, no further changes occur if the original lotka-Volterra model reaches (0,0). However, if we reintroduce, let's say, ten prey and one predator into the system. We will move away from this steady state and move on to the next one if any. Making (0,0) an unstable steady state. We can consider the steady state of the equilibrium the system is working towards. It will most likely never reach this point or need infinite time, given that the rate of change will decrease as its gets closer to the steady state [16].

To find the steady state, we must first find our critical points. These are the point where our system has o change in time. In other words:

$$\Delta x = 0$$

$$\Delta y = 0$$

When looking at our ODEs, there are two scenarios that give us this:

$$x = 0$$

$$y = 0$$

Or

$$(a - h - by) = 0$$

$$(cx - (d + h)) = 0$$

This will yield:

$$x = \frac{d+h}{c}$$

$$y = \frac{a-h}{b}$$

This gives us:

$$(0, 0) \quad (\frac{d+h}{c}, \frac{a-h}{b})$$

These are the steady states. The fixed points at which change will no longer occur. However, their stability, i.e., what will happen when we get near these points, can be calculated by finding their eigenvalues and eigenvector. This can be done by stating the problem as an eigenvalue problem [18]:

$$(A - \lambda I)v = 0$$

- A is the jacobian matrix
- λ is a scaler
- I is the identitymatrix
- v is the eigenvector

The jacobian matrix needed can be calculated by inserting the ODE's into this matrix:

$$\begin{bmatrix} \frac{f_1}{\delta x} & \frac{f_1}{\delta y} \\ \frac{f_2}{\delta x} & \frac{f_2}{\delta y} \end{bmatrix}$$

For our model this will give us:

$$\begin{bmatrix} a - h - by & ax - hx - bx \\ cy - dy - hy & cx - d - h \end{bmatrix}$$

Since our eigenvector is a non-zero vector, we can find the eigenvalues by taking the determinate of our matrix:

$$det\,(A - \lambda I) = 0$$

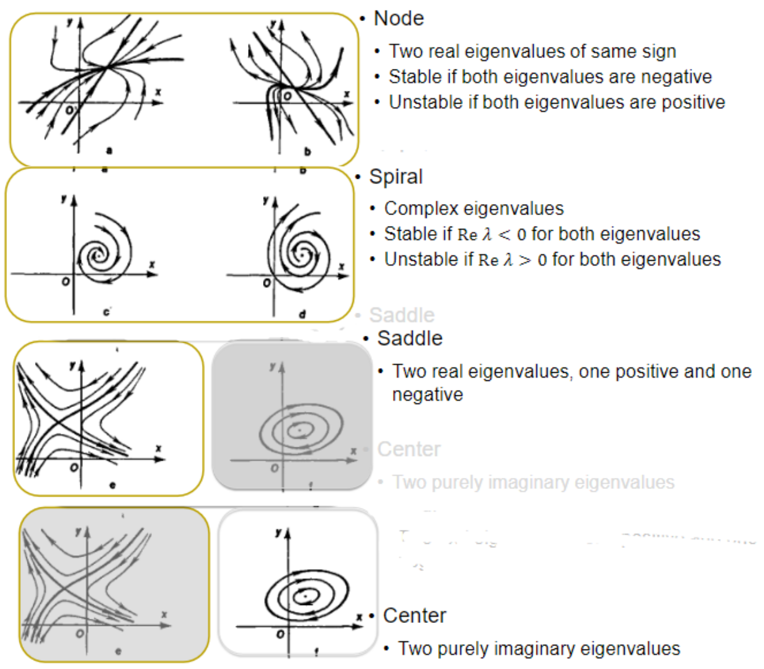
When we add this all together we get:

$$\left|\begin{bmatrix} (a - h - by) - \lambda & ax - hx - bx \\ cy - dy - hy & (cx - d - h) - \lambda \end{bmatrix}\right|$$

Now we can solve for λ:

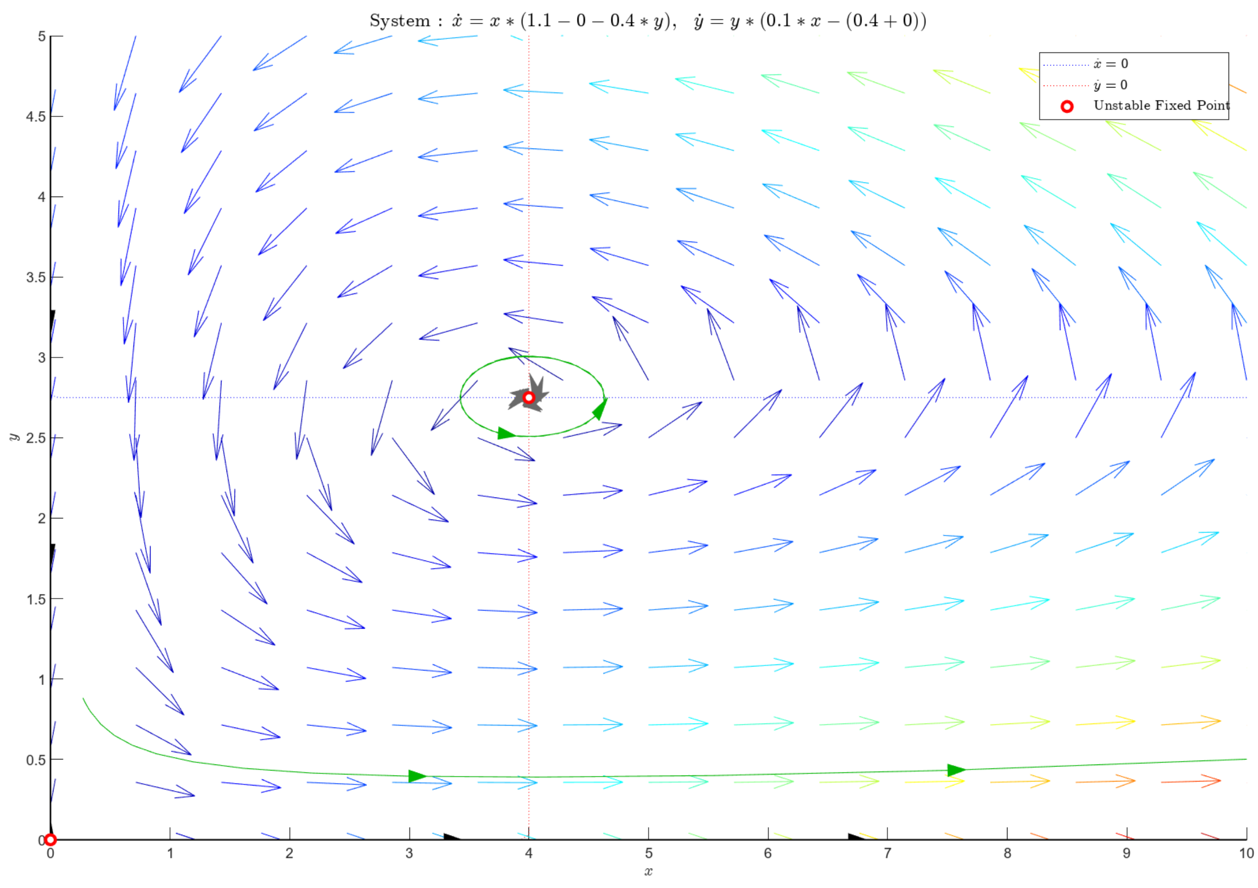
$$((a - h - by) - \lambda) \times ((cx - d - h) - \lambda) - (ax - hx - bx) \times (cy - dy - hy) = 0$$

When we have found our eigenvalues, we can determine if our steady states are stable by plotting them into a phase plot, we can also use this diagram to read out what type of phase plot we will get [15].


• Node
• Two real eigenvalues of same sign
• Stable if both eigenvalues are negative
• Unstable if both eigenvalues are positive
• Spiral
• Complex eigenvalues
• Stable if Re λ < 0 for both eigenvalues
• Unstable if Re λ > 0 for both eigenvalues
• Saddle
• Two real eigenvalues, one positive and one negative
• Center
• Two purely imaginary eigenvalues

We will now look at the phase plots for our prey-predator model and see how harvesting will affect our model. Finally, we will compare this to the original model, see if there are any major changes, and see what this tells us about the system.

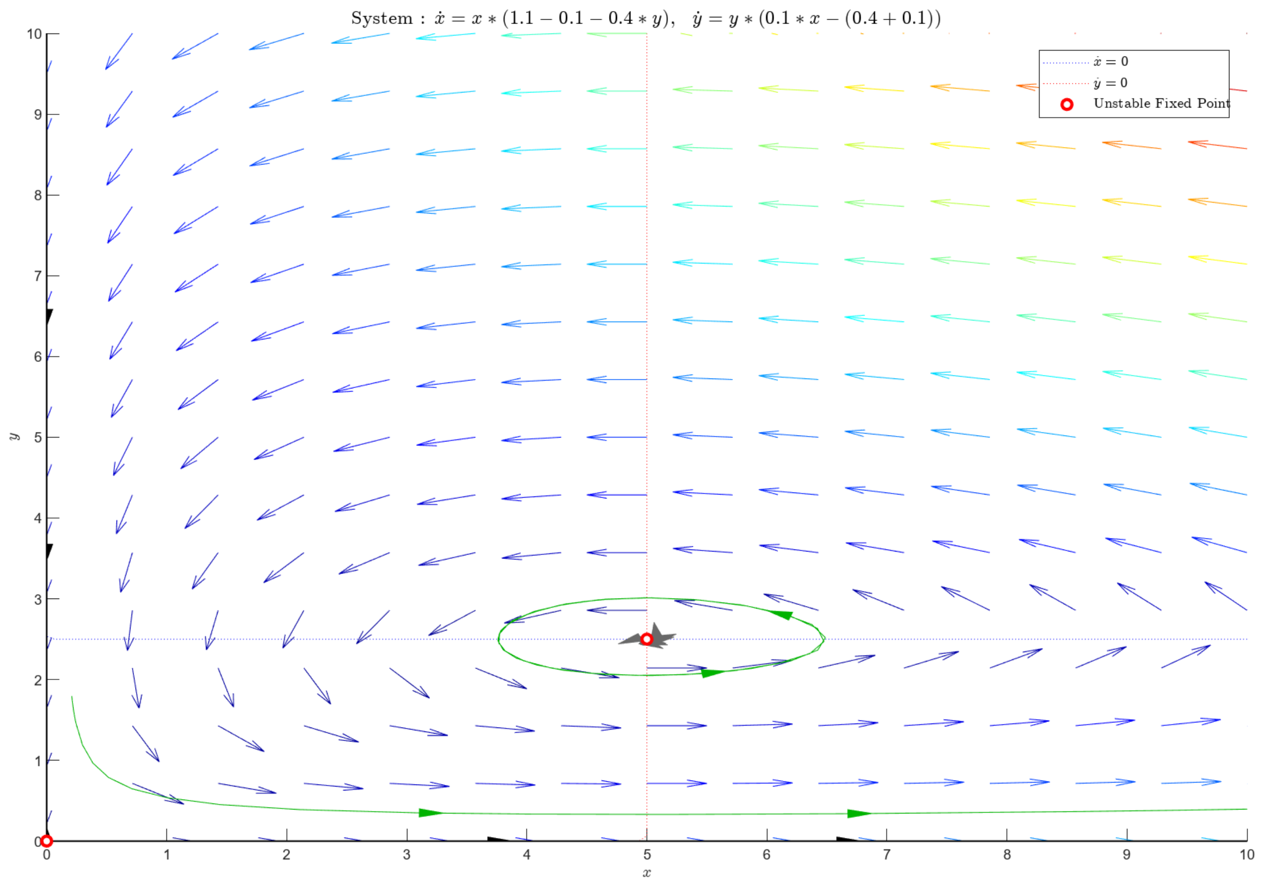
Phase plot for the original Lotka Volterra model - harvesting is 0



Here, we can see that the fixed points are at (0,0) and (4,2.75). The calculated eigenvalues are

- (0,0) - $\lambda_1 = \frac{-2}{5}$, $\lambda_2 = \frac{11}{10}$
- (4,2.75) - $\lambda_1 = -\left(\frac{1}{5}\left(\sqrt{11} \times 1i\right)\right)$, $\lambda_2 = \left(\frac{1}{5}\left(\sqrt{11} \times 1i\right)\right)$

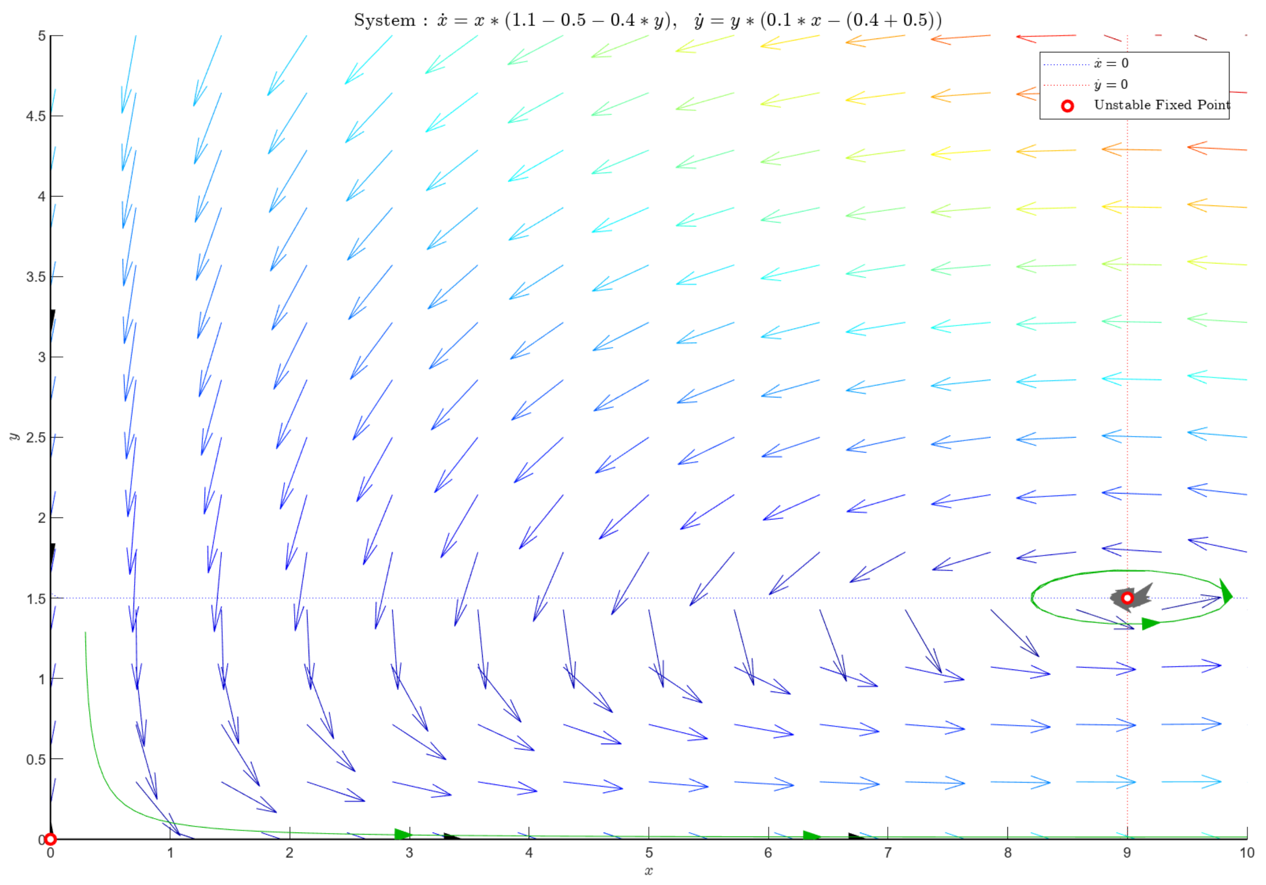
Phase plot for Lotka Volterra model with harvesting - harvesting rate is 0.1



Here, we can see that the fixed points are at (0,0) and (5,2.5). The calculated eigenvalues are:

- (0,0) - $\lambda_1 = \frac{-1}{2}, \quad \lambda_2 = 1$
- (5,2.5) - $\lambda_1 = -\left(\frac{1}{2}\left(\sqrt{2} \times 1i\right)\right), \quad \lambda_2 = \left(\frac{1}{2}\left(\sqrt{2} \times 1i\right)\right)$

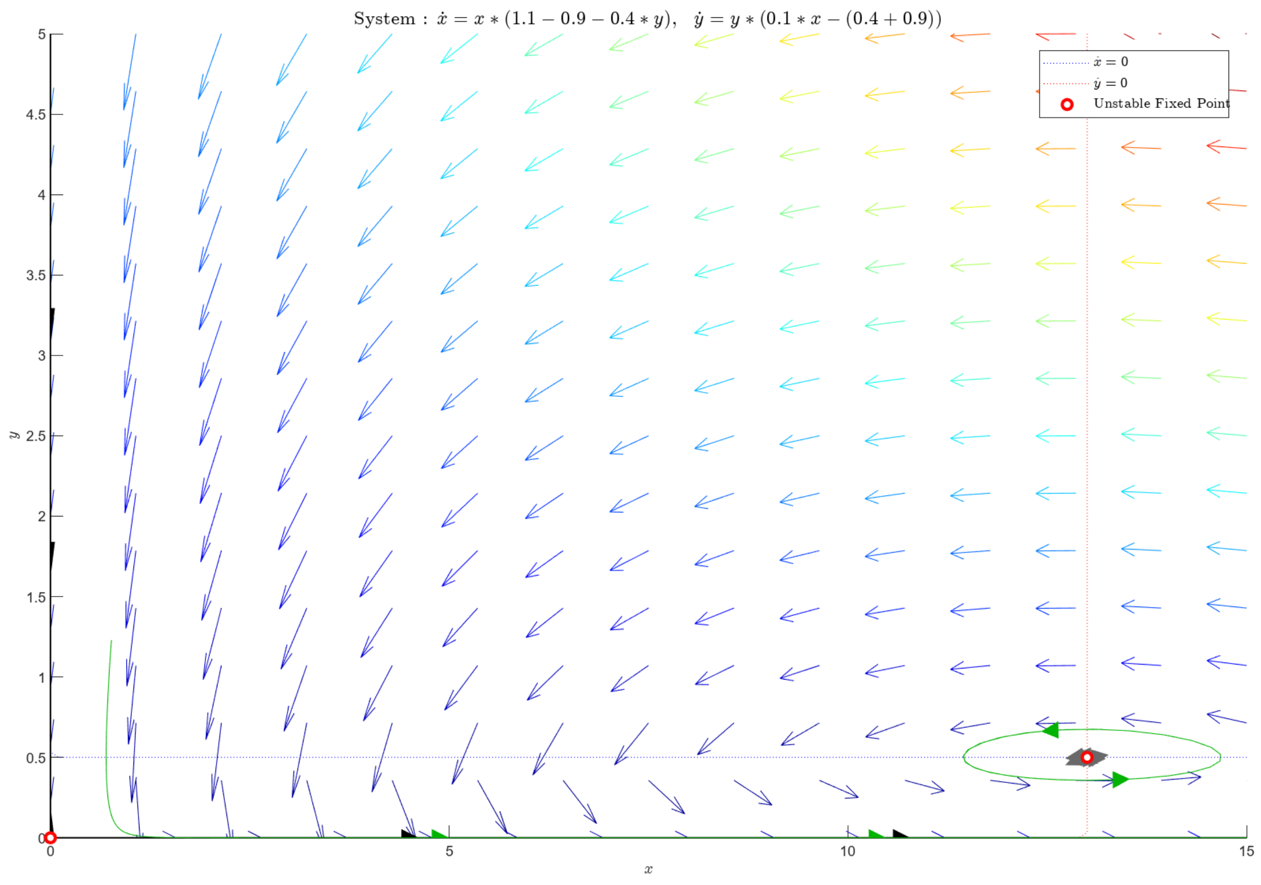
Phase plot for Lotka Volterra model with harvesting - harvesting rate is 0.5



Here, we can see that the fixed points are at (0,0) and (9,1.5). The calculated eigenvalues are:

- (0,0) - $\lambda_1 = \frac{-9}{10}$, $\lambda_2 = \frac{3}{5}$
- (9,1.5) - $\lambda_1 = -(\frac{1}{10}(\sqrt{6} \times 3i))$, $\lambda_2 = (\frac{1}{10}(\sqrt{6} \times 3i))$

Phase plot for Lotka Volterra model with harvesting - harvesting rate is 0.9



Here, we can see that the fixed points are at (0,0) and (13,0.5). The calculated eigenvalues are:

- (0,0) - $\lambda_1 = \frac{-13}{10}, \quad \lambda_2 = \frac{1}{5}$
- (13,0.5) - $\lambda_1 = -\left(\frac{1}{10}\left(\sqrt{26} \times 1i\right)\right), \quad \lambda_2 = \left(\frac{1}{10}\left(\sqrt{26} \times 1i\right)\right)$

We can see from our different phase plots that we get two steady states, as we mentioned previously.
We always have one at (0,0), which is always unstable. The reason is that as soon as one of our equations gets closer to zero, it will start impacting the other equation.

$$x' = x(a - h - by)$$

$$y' = y(cx - (d + h))$$

As can be seen by the birth rate of predators and the death rate of prey being linked to the other. As explained in the earlier chapter. As the predators grow, the number of prey will decrease. However, at some point, that will affect the predators as well, and they will decrease. Giving rise again to the prey, which will then increase. We will see how this manifests itself over time in the next chapter.

For our second fixed point, we can see that a circle point develops, and we get a continuous loop that it keeps. This, again, is a function of the model where prey and predators are linked. This point is neither stable nor unstable but can be classified as stable. This is because it will oscillate around this point. Predators will decrease the number of prey, and prey will diminish, leading to fewer predators, which will give rise to more prey, which will continue back and forth. This might be the origin of the term “circle of life.”

While looking at the different plots, we can see that when we only adjust our harvesting rate equally for both equations, the second fixed point favors more prey and fewer predators. This might be counterintuitive, given that we would assume that predators would always be favored; however, if we look at how the model is set up. We can see that the death rate of predators is amplified by this harvesting rate, which makes it so that they die at a higher rate, unlike prey which is a linear deduction over time.

In the next chapter, we will take a closer look at how these phenomena we discussed above appear when we look at the model over time.

Numerical methods and schemes

A numerical method or scheme is a tool used to solve numerical problems [1]. It is deployed when there is no easy way or way of solving a problem analytically. These numerical methods give us an approximation of what the solution could be. They are not exact, but our best guess to a solution. Numerical methods are implemented using computers. The method is coded with Matlab, python, or any other qualified programming language. The model equation is then fed, and different values for the constants in the equation are chosen. The result is a graphical representation of the change over time given the chosen constants [2].

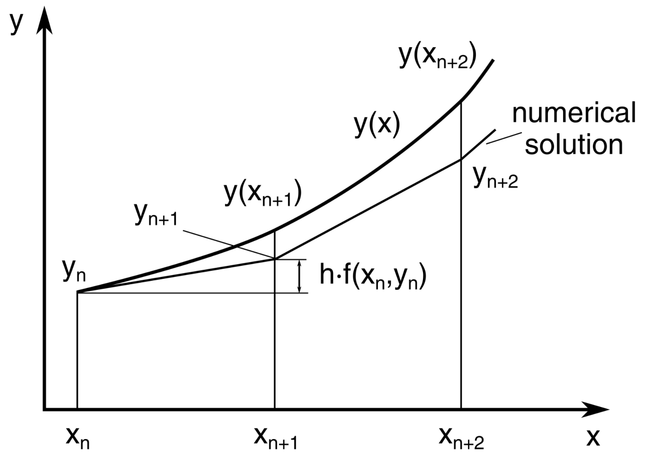
This section will go through the different numerical methods used in this report. And we will, later on, see how they compare to each other when solving our model.

Eulers Method

This might be the most known numerical method and the simplest. It takes in our problem, assuming it's presented as a Taylor series [4]. This allows us to take the first to terms and write it as such:

$$y(x + h) = y(x) + hf(x, y)$$

This then allows us to calculate y_n at h step way. As can be seen from the picture below [5]:



The picture above illustrates how by using this numerical method, we know can start to plot a graph that shows us how over time, the series will evolve. Mathematically this means that the first term:

$$y_1 \approx y_0 + hf(x_0, y_0)$$

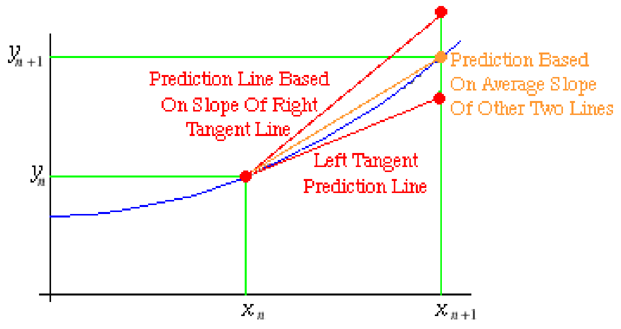
Where y_0 and x_0 are known initial values. y_1 is the next step and we use this to create our next point:

$$y_2 \approx y_1 + hf(x_1, y_1)$$

We do this for how many steps, h we have.

Heuns Method

A step up from Euler's simple method is the Heun method. Named after Karl Heun. It's usually referred to as an improved Eulers method. The main thing it improves is its accuracy. While the Eulers method tries to calculate the slope between two points by using the line tangent, which works for smaller problems, over a larger area, there will be errors. Heun's method tries to mitigate this problem by calculating the average of two slopes instead of just one, like Euler. By having a tangent line on both sides of the curve, we can calculate the average of these two points. The problem of overestimating or underestimating the next point is used to our advantage. By calculating on both sides of the tangent, we can now get a better approximation, given we have overestimated and underestimated the next point on the curve [9]. As seen in this graph, we can see how the Heun method improves upon the Euler method [9]:


y_{n+1}
y_n
Prediction Line Based On Slope Of Right Tangent Line
Prediction Based On Average Slope Of Other Two Lines
Left Tangent Prediction Line
x_n
x_{n+1}

The mathematical expression of Heuns method is this [6]:

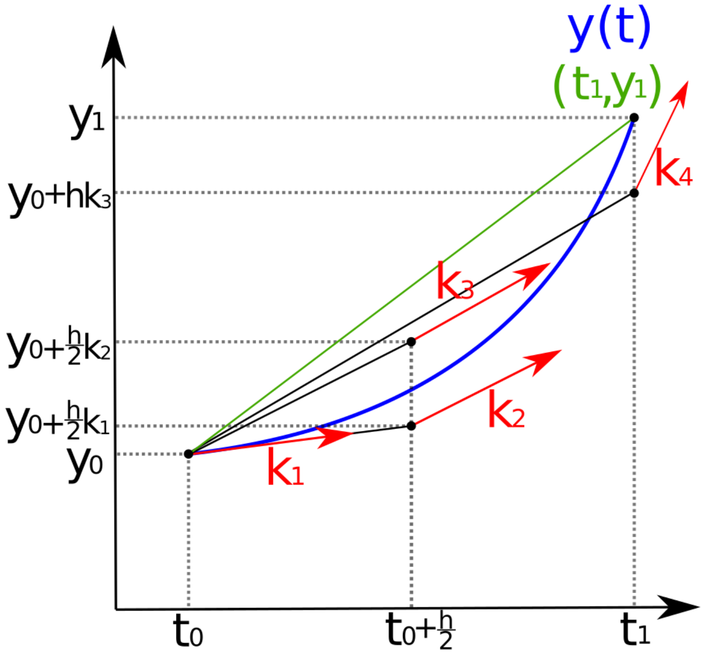
$$y^{p}_{n+1} = y_n + \; hf(x_n, y_n)$$

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) \; + \; f(x_{n+1}, y^{p}_{n+1})]$$

The idea follows the same steps as Euler; we keep calculating the next step on the curve. However, as mentioned, by calculating for a second tangent line, we get a more efficient and accurate graphical representation.

Runge-Kutta Method

As the need to solve more complex problems arise, we also need more powerful tools. Both Euler and Heun are prone to eros. The Heun less so than Euler. Building on top of these, we have Runge-Kutta. The Runga-Kutta method, otherwise known as RK4, introduces a half-step. This gives us more points to calculate when plotting our graph. This makes it so that we get a more accurate estimation of the next point. One might think that decreasing the step count in Euler and Heun would give the same result; however, this is a more efficient way of realizing the same result. This also minimizes the errors compared to those two other methods. It can be compared to the Trapezoidal rule. By using more steps, we can now get a better read on where the slope of our curve will accrue and adjust our points accordingly [8].



The mathematical formulation of the Runge-Kutta is as follows [12] :

$$
\begin{aligned}
y_{n+1} &= y_n + \tfrac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\
t_{n+1} &= t_n + h \\
k_1 &= f(t_n, y_n), \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right), \\
k_4 &= f\left(t_n + h, y_n + hk_3\right).
\end{aligned}
$$

Here we can see that for each y_{n+1} we calculate multiple steps along the line, giving us the diagram we see above.

Results

In this chapter, we will look at the results of implementing the different methods. We have used Matlab to generate the solutions. Every method was run with three different harvesting rates, and the other constants were the same for every scenario. We wish to observe how the different methods perform and what they tell us about our model. Specifically how adding this harvesting rate changes things. To get a better overview of this, we also ran each method once without the harvesting rate to compare.

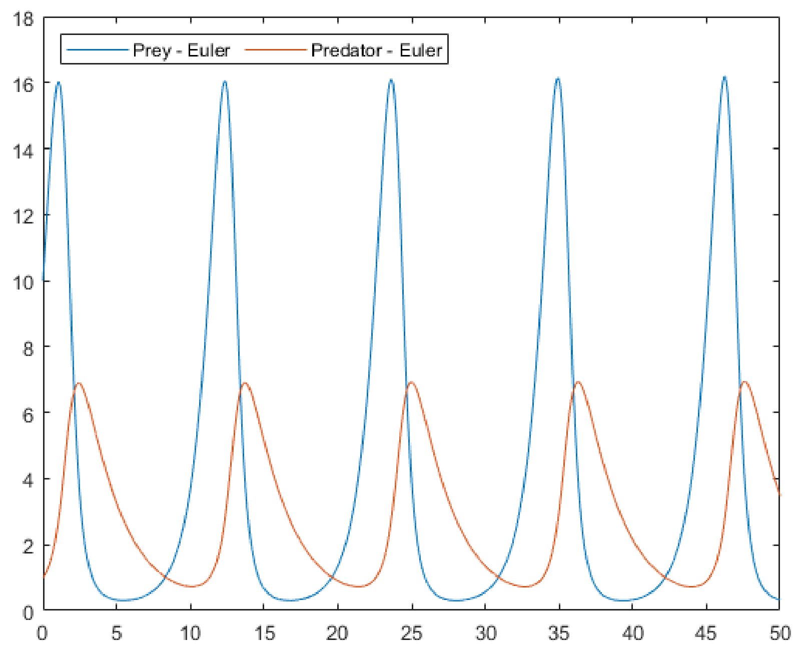
The three different scenarios chosen were:

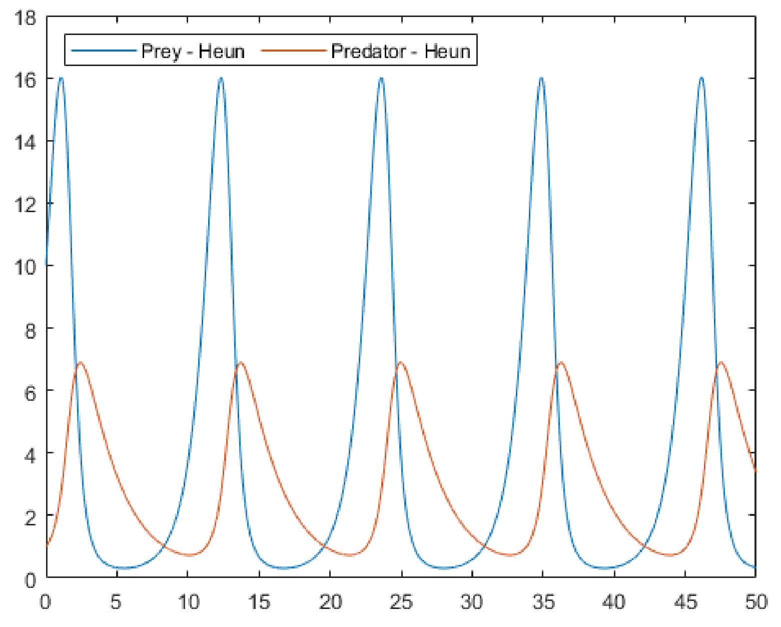
$h = 0.1,\ 0.5\ and\ 0.9$

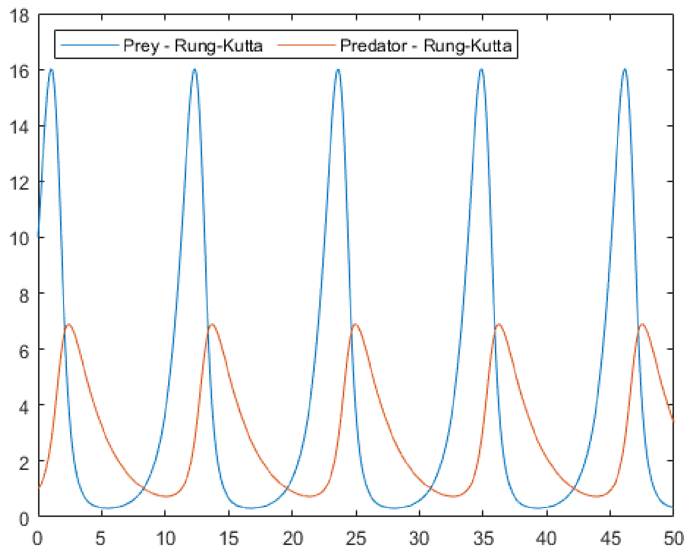
All the other constants were set as seen below:

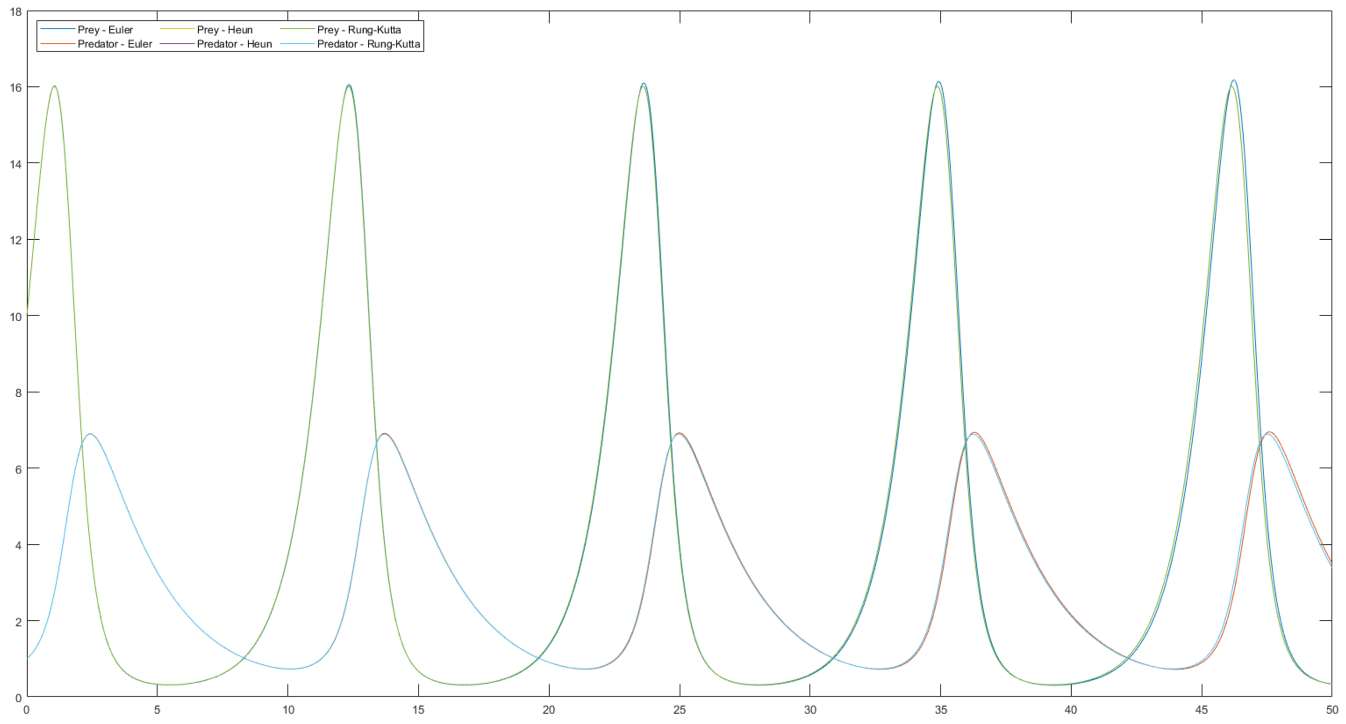
- $a = 1.1$
- $b = 0.4$
- $c = 0.1$
- $d = 0.4$

Scanario 0 - harvesting rate 0

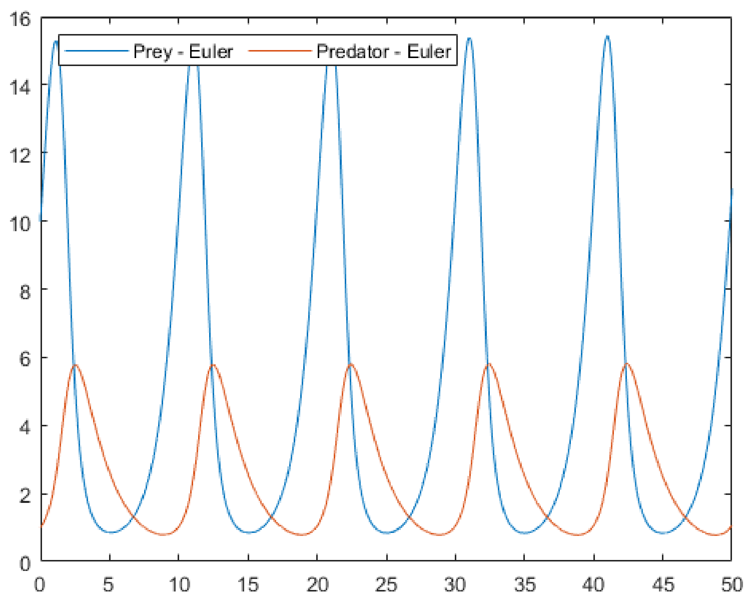


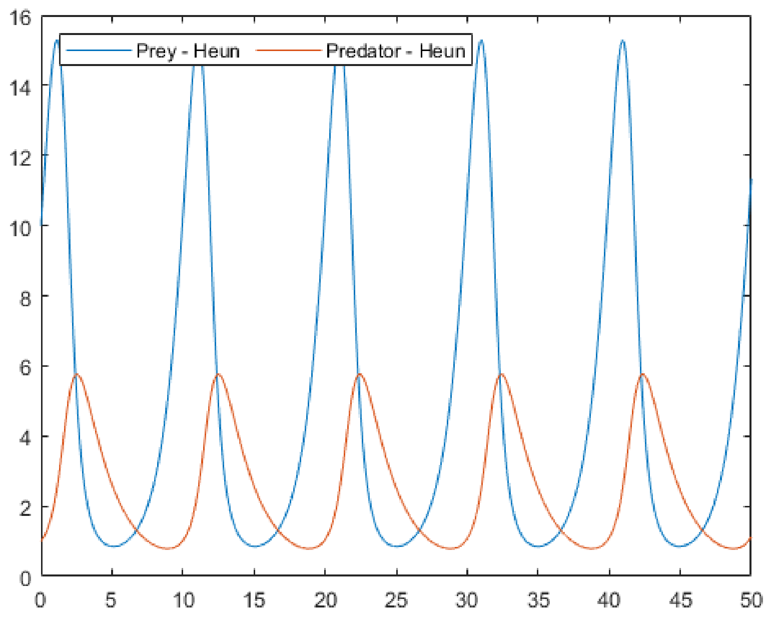


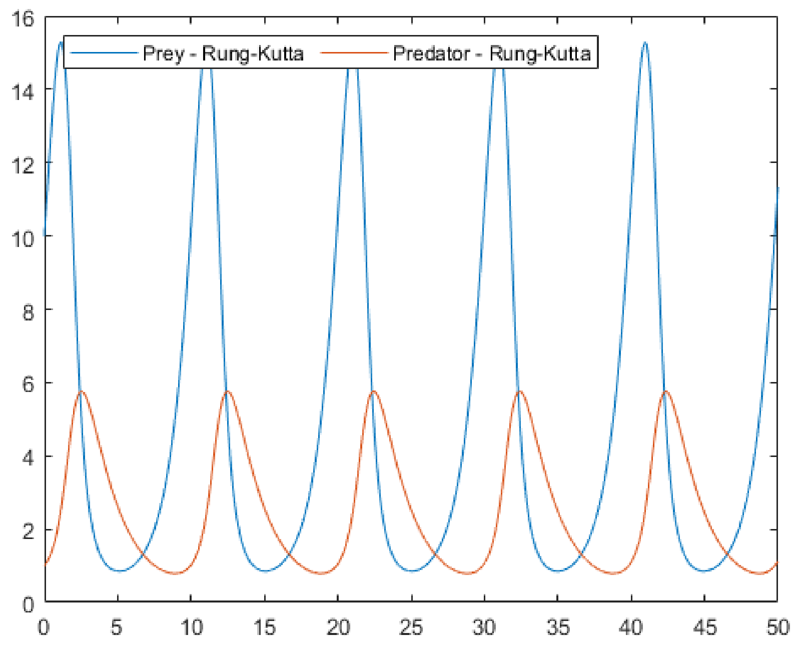

Prey - Rung-Kutta
Predator - Rung-Kutta

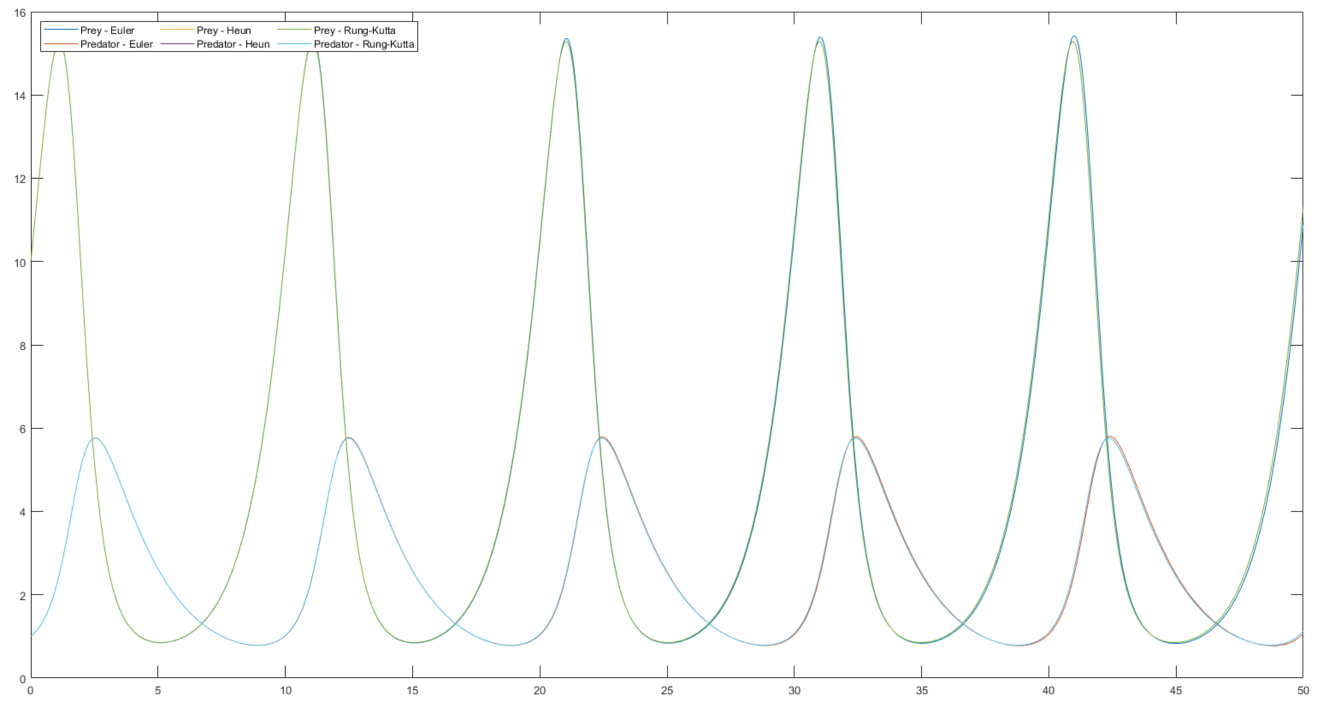

Prey - Euler
Predator - Euler
Prey - Heun
Predator - Heun
Prey - Rung-Kutta
Predator - Rung-Kutta

Scanario 1 - harvesting rate 0.1

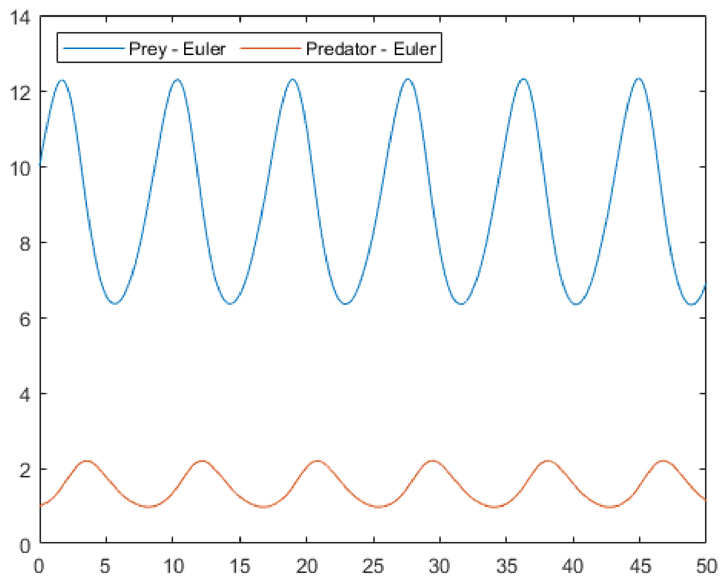


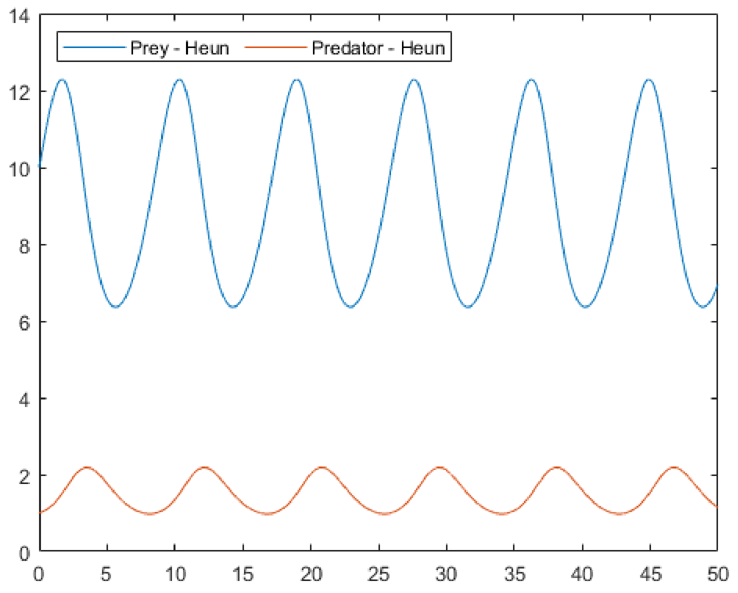



Prey - Rung-Kutta
Predator - Rung-Kutta

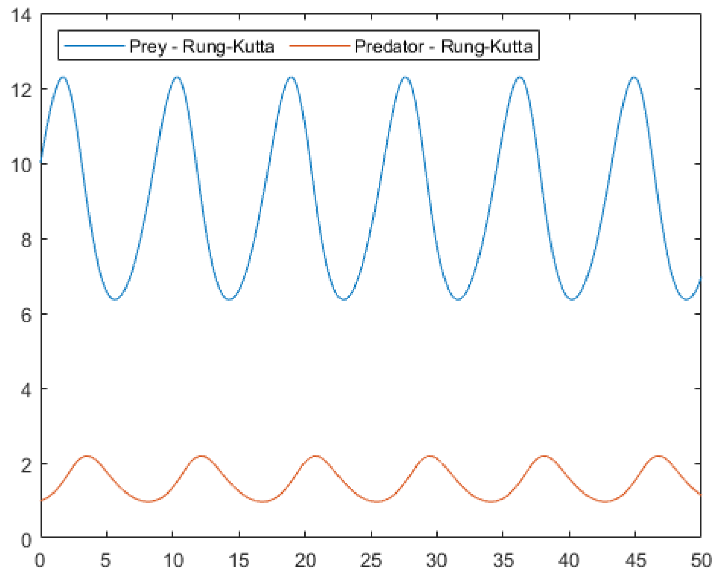

Prey - Euler
Predator - Euler
Prey - Heun
Predator - Heun
Prey - Rung-Kutta
Predator - Rung-Kutta

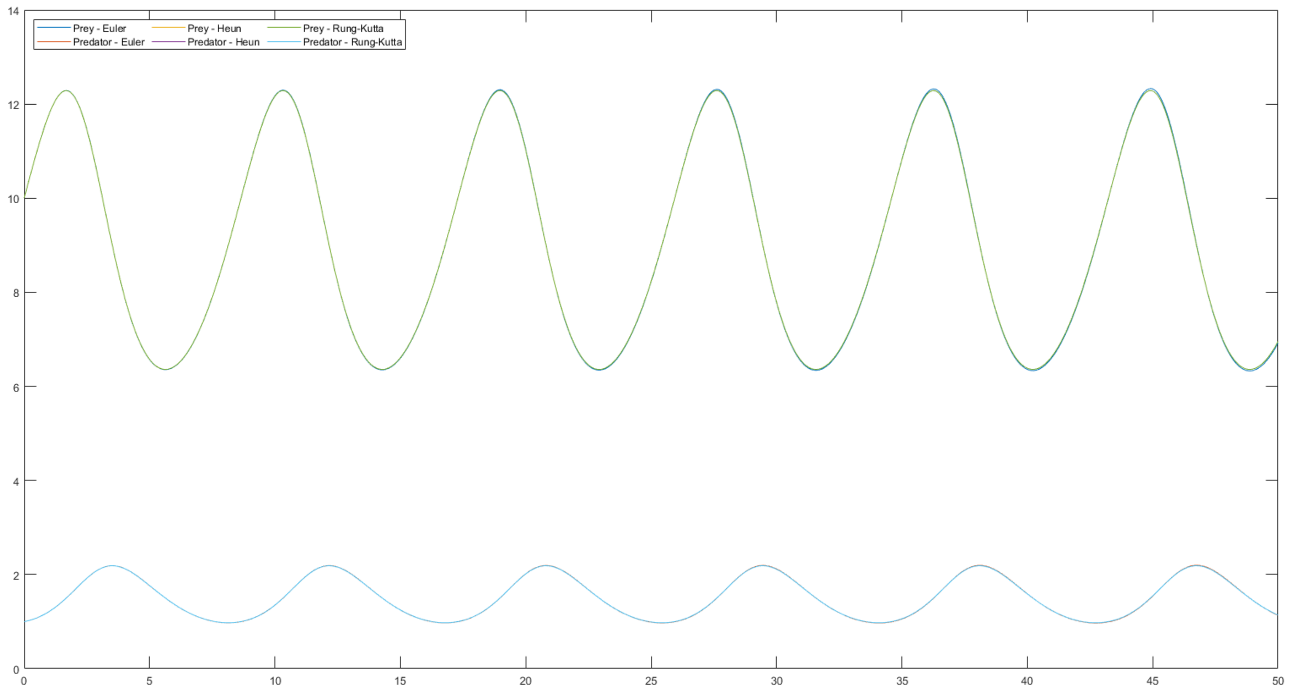
Scanario 2 - harvesting rate 0.5





Scanario 3 - harvesting rate 0.9

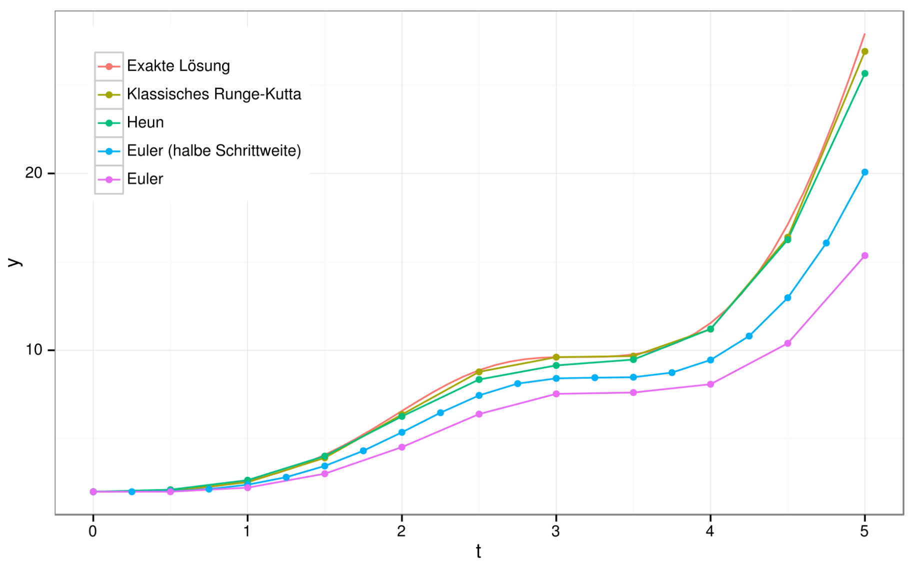




As can be seen from the graphs above. We get varying results as we increase the harvesting rate. Having a harvesting rate below the death rate of our predators keeps our predators in line with the prey. The prey reaches a higher high than the predators, but the lows of the prey go below the number of predators. When the rate of harvesting passes our death rate, we see that now the prey never drops to the level of predators and always outnumbers them.

A second thing to note is that as harvesting increases, the number of prey and predators slowly drop. For example, we can see that the high is around 16 prey and seven predators when the harvesting rate is set to 0; however, when we get to our last scenario and set the harvesting rate to 0.9, the values have dropped to around 12 and 2 for each. This makes sense, given that we are removing a portion of the population, which will keep the population lower.

Lastly, what can be said about our numerical solutions is that the error rate for the different numerical methods shows up as we move in time. The Euler methods showed a degree of decay in their accuracy, but Heun and Rung-Kutta were very similar. However, it's suspected and expected that, given enough time, the error would be more visible in our graph for both Euler and Heun. The difference in their accuracy is something that has been documented before [13]. The graph below shows this very clearly [14]:



Conclusion

In this report, we have a look at how our prey-predator model works. First, we have seen the numerical analysis, giving us the fixed points, and see how the harvesting rate impacts our steady states and that harvesting favors prey more than predator. Then, we ran our system through different numerical methods and simulated how the prey-predator interaction goes over time. We saw that the different numerical methods were somewhat similar; however, the error rate starts to show, given a long enough time. Here we also learned that the harvesting rate has a huge impact on our system, given everything else being equal, as seen from the graphs.

I think what we can take away from this project is that the Lotka-Volterra model with the harvesting rate shows how a minor change in a model can quickly change the outcome. We saw this when we compared the original model with the variation we worked on.

We can also take away that not all numerical methods are created equal; under some circumstances, they don't all hold the same truthfulness in their answer. Therefore, the need for efficient and more accurate methods as systems get more and more complex is necessary.

References

[1]A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*. Springer Science & Business Media, 2006.

[2]“What is a numerical method?,” *StudySmarter US*. https://www.studysmarter.us/explanations/math/pure-maths/numerical-methods/ (accessed Nov. 28, 2022).

[3]“Differential Equations,” *Euler’s Method*. https://tutorial.math.lamar.edu/classes/de/eulersmethod.aspx (accessed Nov. 28, 2022).

[4]M. Bourne, “11. Euler’s Method,” *a numerical solution for Differential Equations*. https://www.intmath.com/differential-equations/11-eulers-method-des.php (accessed Nov. 28, 2022).

[5]freeCodeCamp.org, “Euler’s Method Explained with Examples,” *freeCodeCamp.org*, Jan. 26, 2020. Accessed: Nov. 28, 2022. [Online]. Available: https://www.freecodecamp.org/news/eulers-method-explained-with-examples/

[6]NTNU, “Numerical Methods for Engineers,” *NTNU.no*. https://folk.ntnu.no/leifh/teaching/tkt4140/._main015.html#mjx-eqn-2.76 (accessed Nov. 28, 2022).

[7]A. J. Davies, “Runge-Kutta Numerical Integration of Ordinary Differential Equations in Python,” *Geek Culture*, Jun. 13, 2022. Accessed: Nov. 28, 2022. [Online]. Available: https://medium.com/geekculture/runge-kutta-numerical-integration-of-ordinary-differential-equations-in-python-9c8ab7fb279c

[8]tok.wiki, “วิธี Runge–Kutta วิธี Runge–Kuttaและวิธี Runge–Kutta ที่ชัดเจน,” *tok.wiki*. Accessed: Dec. 01, 2022. [Online]. Available: https://hmong.in.th/wiki/Runge-Kutta_method

[9]Deltacollege, “Numerical Methods--Heun’s Method,” *Calculuslab*. http://calculuslab.deltacollege.edu/ODE/7-C-2/7-C-2-h.html (accessed Dec. 01, 2022).

[10]Low-dimensional Energy Balance Models, “lowEBMs.Packages.RK4 — Low-dimensional Energy Balance Models documentation,” *Lowebms*. https://lowebms.readthedocs.io/en/latest/code/rk4.html (accessed Dec. 01, 2022).

[11] https://i.stack.imgur.com/fZh9b.png (accessed Dec. 01, 2022).

[12]poisson, “The Runge-Kutta method for a system of equations,” *Mathematics Stack Exchange*. https://math.stackexchange.com/questions/2603295/the-runge-kutta-method-for-a-system-of-equations (accessed Dec. 01, 2022).

[13]NTNU, “Numerical Methods for Engineers,” *NTNU*. https://folk.ntnu.no/leifh/teaching/tkt4140/._main024.html (accessed Dec. 02, 2022).

[14]Contributors to Wikimedia projects, “Runge–Kutta methods,” *Wikipedia*, Oct. 15, 2022. https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods (accessed Dec. 02, 2022).

[15]Oslomet, “Lecture notes week 34,” *Canvas*, 2022. https://oslomet.instructure.com/courses/24252/files/folder/Lecture%20notes?preview=2811036 (accessed Dec. 02, 2022).

[16]M. Kumar, “Phase Portrait Plotter,” *MATLAB Central*, Mar. 28, 2021. https://se.mathworks.com/matlabcentral/fileexchange/81026-phase-portrait-plotter (accessed Dec. 02, 2022).

[17]F. Hoppensteadt, “Predator-prey model,” *Scholarpedia*, vol. 1, no. 10, Oct. 2006, doi: 10.4249/scholarpedia.1563.

[18]mathsisfun, “Eigenvector and Eigenvalue,” *Mathsisfun*. https://www.mathsisfun.com/algebra/eigenvalue.html (accessed Dec. 02, 2022).

Appendix: code

Main code

```
time=0:0.001:50; %change time,step,limit
yo=[10,1]; %change prey,preadtor initial condition
Euleroutput=Euler(time,yo,@LotkaVolterra); %implement numerical method
Heunoutput = Heun(time, yo,@LotkaVolterra) %implement numerical method
Kuttaoutput = RungeKutta4(time,yo,@LotkaVolterra) %implement numerical method
plot(time,Euleroutput) %plot graph
plot(time, Heunoutput) %plot graph
plot(time,Kuttaoutput) %plot graph
plot(time,Euleroutput,time,Heunoutput,time,Kuttaoutput) %plot graph
legend({'Prey - Euler','Predator - Euler'},'Location','northwest','NumColumns',3)
legend({'Prey - Heun','Predator - Heun'},'Location','northwest','NumColumns',3)
legend({'Prey - Rung-Kutta','Predator - Rung-Kutta'},'Location','northwest','NumColumns',3)
legend({'Prey - Euler','Predator - Euler','Prey - Heun','Predator - Heun','Prey - Rung-Kutta','Predator -
Rung-Kutta'},'Location','northwest','NumColumns',3)
```

Equation

```
function Y = LotkaVolterra(t,X)
x=X(1);
y=X(2);
% Parameters:
a=1.1;
b=0.4;
c=0.1;
d=0.4;
h=0.4;
Y=zeros(2,1);
Y(1)=x*(a-h-b*y);
Y(2)=y*(c*x - (d + h));
end
```

Functions

Euler

```
function Y = Euler( t,yo,F )
Lt=length(t);
h=t(2)-t(1);
Y=zeros(length(yo),Lt);
Y(:,1)=yo;
for k=2:Lt
  Y(:,k)=Y(:,k-1)+h*F(t(k-1),Y(:,k-1));
end
end
```

Heun

```
function Y = Heun( t,yo,F )
Lt=length(t);
h=t(2)-t(1);
Y=zeros(length(yo),Lt);
Y(:,1)=yo;
for j=1:Lt-1
  k1=F(t(j),Y(:,j));
  Yeuler=Y(:,j)+h*k1;
  k2=F(t(j+1),Yeuler);
  Y(:,j+1)=Y(:,j)+(h/2)*(k1+k2);
end
end
```

Runge-Kutta

```
function Y = RungeKutta4( t,yo,F )
Lt=length(t);
h=t(2)-t(1);
Y=zeros(length(yo),Lt);
Y(:,1)=yo;
for j=1:Lt-1
  k1=h*F(t(j),Y(:,j));
  k2=h*F(t(j)+h/2,Y(:,j)+k1/2);
  k3=h*F(t(j)+h/2,Y(:,j)+k2/2);
  k4=h*F(t(j)+h,Y(:,j)+k3);
  Y(:,j+1)=Y(:,j)+(1/6)*(k1+2*k2+2*k3+k4);
end
end
```

Jacobian and Eigenvalue

```
syms x y;
format longEng;
a=1.1;
b=0.4;
c=0.1;
d=0.4;
h=0.9;
F = [a*x-h*x-b*y*x,c*x*y-d*y-h*y]
dx = a*x-h*x-b*y*x;
dy = c*x*y-d*y-h*y;
[xcr,ycr]=solve(dx,dy)
C1 = [xcr(1) ycr(1)]
C2 = [xcr(2) ycr(2)]
v = [x y]
J = jacobian(F,v)
C1 = [xcr(1) ycr(1)] %critical point 1
C2 = [xcr(2) ycr(2)] %critical point 2
j1 = subs(J,v,C1) %Jacobian Matrix (critical point 1)
j2 = subs(J,v,C2) %Jacobian Matrix (critical point 2)
e1 = eig(j1) %eigenvalues (critical point 1)
e2 = eig(j2) %eigenvalues (critical point 2)
```

Appendix: Tools

- MatLab - https://se.mathworks.com/products/matlab.html
- Phase Portrait Plotter - https://se.mathworks.com/matlabcentral/fileexchange/81026-phase-portrait-plotter