

Investigating Predator - Prey Model of Lynx and Hare

Umay Gokturk, David Ou, Martyna Wojciechowska

*

1 Introduction

In this data-driven model project, we construct a Lotka-Volterra predator-prey inverse problem to analyze the dynamical relationship between the Lynx and Hare in historical population data in Canada.

We first introduce the Lotka-Volterra equations in Section 2. We also visualize the model using fixed parameters. To bring in more realistic and complex examples, we find a dataset with historical data about Lynx and Hare population from 1835 to 1935 in Canada (Hundley, 2003). Then, in Section 5, we set up an inverse problem to estimate the model's best-fitting parameters. Our group uses a loss function that computes the weighted sum of squared errors to measure the degree of misfit from real data. By minimizing the weighted sum of squared errors from the loss function, we arrive at a set of optimal parameters for the Lynx and Hare predator-prey model. To enhance realism, we extend the model with damping terms, accounting for intra-species competition and environmental constraints.

In Section 6, we compare our fitted models with real-world data, and evaluate the model's predictive power and discuss limitations, such as capturing extreme population fluctuations.

2 Lotka–Volterra predator–prey model

The Lotka–Volterra predator–prey model, also known as The Lotka–Volterra equations, is a pair of first-order nonlinear differential equations. They describe the dynamics of two populations, the predators and the prey the feed on. Both populations change according to these equations (Anisiu, 2014):

$$\begin{aligned}\frac{dx}{dt} &= \alpha x - \beta xy, \\ \frac{dy}{dt} &= -\gamma y + \delta xy\end{aligned}\tag{1}$$

where,

y is the variable for predator population. The $\frac{dy}{dt}$ describes the growth rate of that population. Similarly, x is the variable describing the prey population, and $\frac{dx}{dt}$ describes its growth rate depending on time t . The coefficient $\alpha > 0$ describes the maximum growth per capita for the prey population. If the population of predators was non-existent, then the growth rate for the prey population would grow proportionally to its size:

$$\frac{dx}{dt} = \alpha x\tag{2}$$

If the population of prey was absent, then the population of predators would decline according to the parameter $\gamma > 0$ such that:

$$\frac{dy}{dt} = -\gamma y \quad (3)$$

According to (Anisiu, 2014), if both prey and predators exist in a given environment, prey population will decline, while predators' grow. The rate of these changes will be proportional to the frequency of encounters of both species.

For $\beta, \gamma > 0$, the term βxy describes the change in prey population due to interactions with predators, while the term γxy represents how the predator population changes due to the same interaction.

2.1 Simple Prey-Predator Model

To plot the equations, we have used some random parameters and initial conditions. We have used $x(0) = 1$ and $y(0) = 0.5$ for the initial conditions and for parameters we have selected $\alpha = 2$ for prey birth rate, $\beta = 1.1$ for predation rate, $\gamma = 1$ for predator death rate, and $\delta = 0.9$ for predator reproduction rate. (Anisiu, 2014)

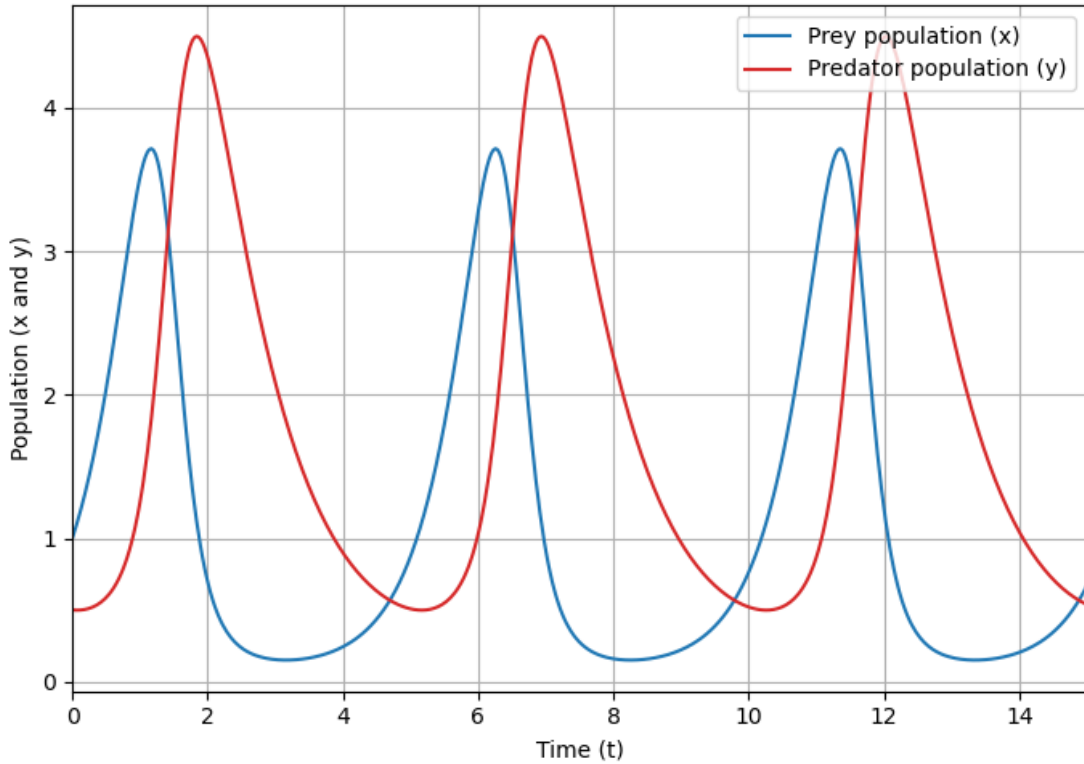


Figure 1: Simulated Prey and Predator model with random parameters, see appendix I

As we can see from Figure 1, both the prey and predator populations keep cycling over time. The prey population increases first, followed by the predator population. Then both drop and repeat the cycle. This kind of behavior shows how the two species depend on each other, and how one population going up or down affects the other.

2.2 Simple Prey-Predator Model with Damping

We can expand the original Lotka–Volterra model to make it more realistic by adding terms that account for how each species might limit its own growth. To represent this limitation in growth we will add $-\lambda x^2$ for the prey and $-\kappa y^2$ for the predator. These new factors represent things like overcrowding or competition for resources within the same species (Anisiu, 2014). So our system becomes:

$$\begin{aligned}\frac{dx}{dt} &= \alpha x - \lambda x^2 - \beta xy, \\ \frac{dy}{dt} &= -\gamma y - \kappa y^2 + \delta xy\end{aligned}\tag{4}$$

When we add these extra terms, we can see that the population cycles will slowly shrink over time instead of repeating forever. Eventually, both populations settle at stable values, which means they will reach an equilibrium state (Anisiu, 2014). Adding the damping factor is important because in real life, predator and prey numbers do not usually oscillate forever, they often stabilize due to limited food, space, or other natural factors. So, this version of the simulation is a better representation of what happens in nature.

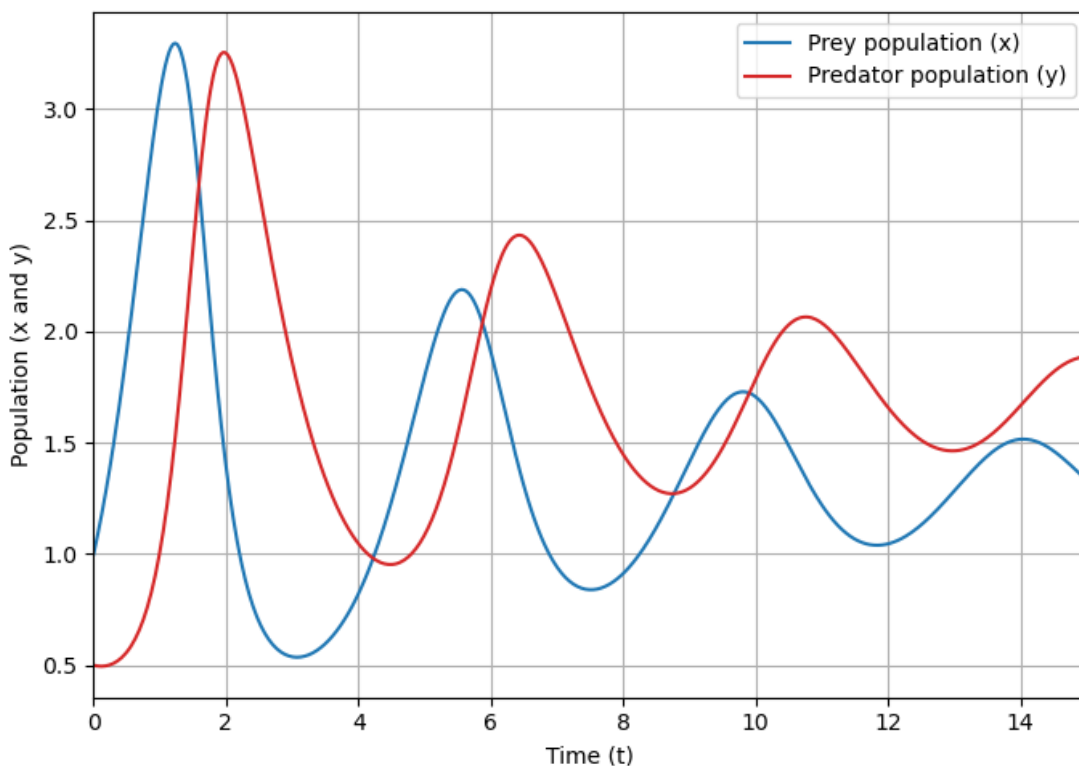


Figure 2: Simulated Prey and Predator model with damping, see appendix II

In this Figure 2, we added damping to make the model more realistic. Unlike the Figure 1, the population cycles get smaller each time until both populations reach and equilibrium. This makes sense in real life, because things like limited food or space would stop the populations from growing and changing in extreme amounts.

3 Data Introduction and Problem Formulation

To bring in a real-world, complex example, we introduce a dataset about the hare and lynx population in Hudson Bay, Canada with data collected between 1845 to 1935. The dataset is accessible here (Hundley, 2003). We will now try to use the Prey-Predator model, to search for parameters using properties of the Inverse Problem introduced in Section 4. Below, as shown on Figure 3, we visualize real data for both populations. We can notice similar patterns and fluctuations as in Figures 1 and 2. We will try to find values for parameters α, β, γ , and δ .

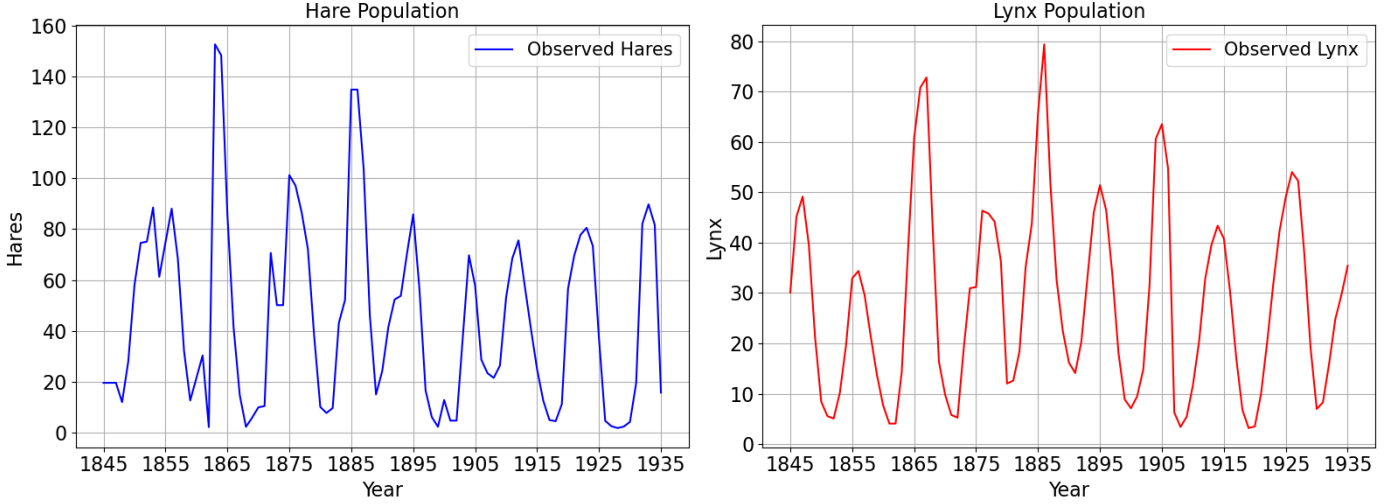


Figure 3: Population of Hare and Lynx, see appendix III

4 Inverse Problem Introduction

For the inverse problem, we observe dataset and try to infer the parameters that fits the dataset where it describes it with minimum error.

In order to fit the Lotka–Volterra model to our historical data of hares and lynxes, we need a way to measure how well our model matches the real populations for any set of parameters. For this, we used a loss function.

After simulating the model with a set of parameters, we get predicted values for both hare and lynx populations at each year. We calculated the loss as the sum of the squared differences between the predicted values and the observed data, across all years.

We defined the loss function as (Alake, 2023):

$$\text{Loss} = \sum_{i=1}^N \left[\left(H_i^{\text{pred}} - H_i^{\text{obs}} \right)^2 + 2 \times \left(L_i^{\text{pred}} - L_i^{\text{obs}} \right)^2 \right] \quad (5)$$

where;

H_i^{pred} is the predicted number of hares at year i ,

H_i^{obs} is the actual (observed) number of hares at year i ,

L_i^{pred} is the predicted number of lynxes at year i ,

L_i^{obs} is the actual (observed) number of lynxes at year i .

We tried to have a weighted loss function by multiplying the squared error for the lynx population by 2 to make the fitting process more sensitive to lynx prediction accuracy (Wikipedia contributors, 2020). Since the predator and prey are both dependent in each other, we needed to give a weight for lynx population because it had fluctuations in our dataset more than the model ODE could handle. When we use squared error, it also helps penalize larger mistakes more heavily, so it gives as less ill-conditioned fitted graphs.

5 Inverse Problem

To fit the Lotka–Volterra model to our dataset, we start by picking initial guesses for the four parameters α, β, γ and δ . These parameters control the dynamics of the hares (prey) and lynxes (predators) based on our ODE. We started with the following initial values: $\alpha = 0.6, \beta = 0.02, \gamma = 0.1$ and $\delta = 0.8$. We chose these values based on trial and error and picked the ones that gave us the possible best fit with our loss function. After we have used these initial parameters to minimize our loss function our best fit was:

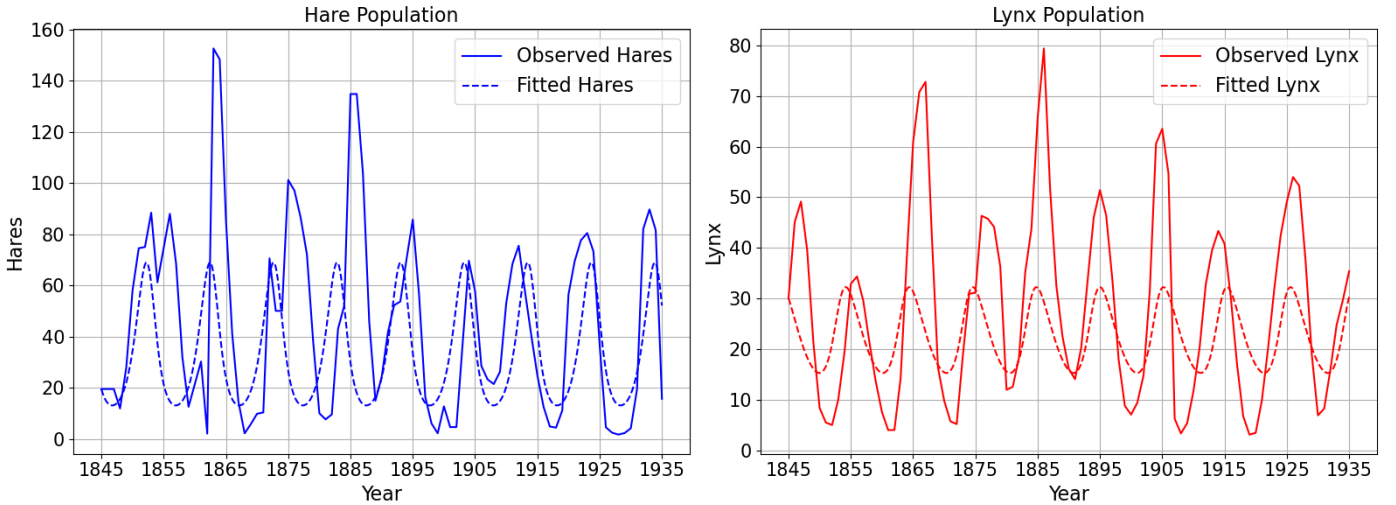


Figure 4: Fitted predator and prey model for Hares and Lynx from dataset, see appendix IV (Saint-Antoine, 2021)

The optimized parameters with the initial guesses we have used were:

$$\alpha = 1.3968, \quad \beta = 0.0614, \quad \delta = 0.0086, \quad \gamma = 0.2901$$

and the minimized loss function value was:

$$\text{final loss} = 178445.1073$$

With these parameters and our choice of loss function we can say that the parameters are fitted nicely for the Hares. Since our ODE equations have cycles which does not have very high inclines

and declines, as seen in Figure 1, when we try to fit our dataset the best optimization cannot capture the very high and low points of the real data set. Also for the Lynx population, we can say that it does not start with a perfect model since for the prey - predator model, we tend to see predators' number to decrease first. But for our model, they increase. Also for predators, we tend to see less fluctuations in the prey-predator model but in our dataset the predators have very high fluctuations. These factors makes our model's best fit not the perfect fit.

For a better and more realistic model, we modified the original equations to include damping terms, which account for limitations in species growth due to overcrowding and resource scarcity. This made our model more realistic since it also can model the oscillations to stabilize over time instead of continuing indefinitely (Errazki, 2022). We then chose initial guesses for the six parameters $\alpha, \beta, \delta, \gamma, \lambda$, and κ . These parameters control the interaction between the hares (prey) and lynxes (predators), along with the self-limiting growth of both species. Our initial values for the model with damping were: $\alpha = 0.8$, $\beta = 0.03$, $\delta = 0.04$, $\gamma = 0.4$, $\lambda = 0.0003$, and $\kappa = 0.0003$. These were chosen based on trial and error and our understanding of how damping should behave in ecological systems. After using these values to minimize the loss function, we obtained our best fit:

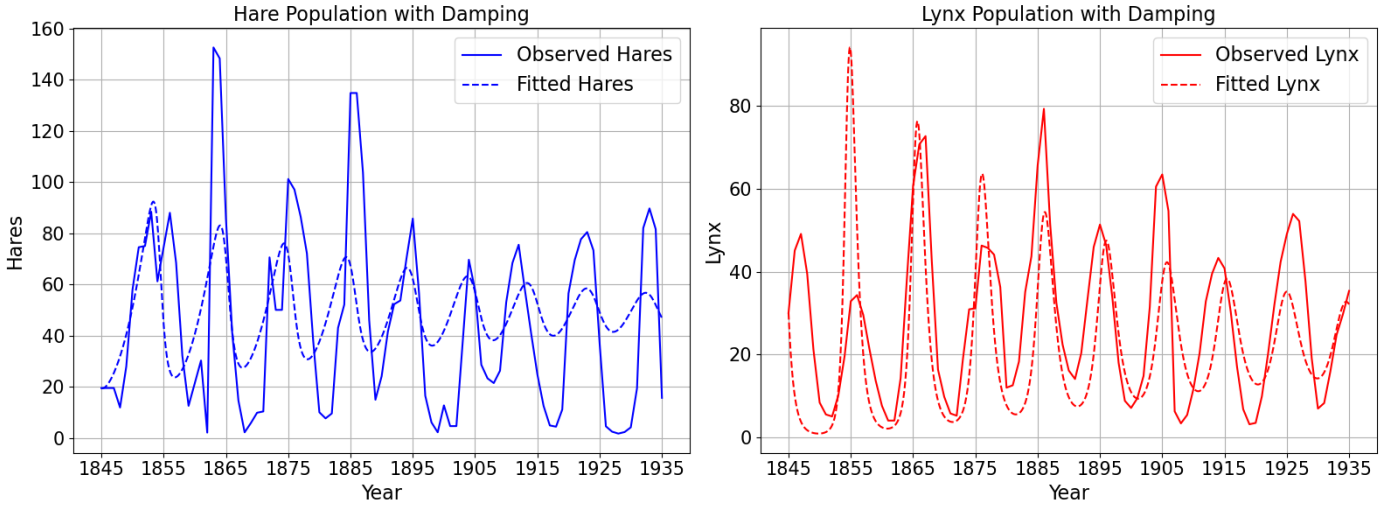


Figure 5: Fitted predator and prey model for Hares and Lynx with damping, see appendix V

The optimized parameters with the initial guesses we have used were:

$$\alpha = 0.2774, \quad \beta = 0.0106, \quad \delta = 0.0375, \quad \gamma = 1.8381, \quad \lambda = 0.000754, \quad \kappa = 0.000144$$

and the minimized loss function value was:

$$\text{final loss} = 142828.0842$$

With the addition of damping terms in our Lotka-Volterra model, we were able to simulate more realistic population dynamics that better resemble the ecological behaviors seen in nature. As shown in Figure 5, the lynx population fitting improved significantly. Our model was able to capture the general timing and shape of the cycles, including their rise and fall patterns. But for hares, our fitting did not show a similar significant improvement with the addition of damping factors. It does not perfectly match the amplitude of the observed peaks and dips, but it still follows the overall pattern

more consistently than in the undamped model. The damping terms help avoid unrealistically sharp oscillations by modeling how intra-species competition limits population growth over time. However, we still observe some differences, particularly in the hare population’s magnitude, which suggests that we need further improvement of the parameters or a more complex model structure to fully capture the behavior of the real data.

5.1 Several different initial guesses

To make our optimization better and minimize the error, we can use several different methods. Firstly, we can try several different initial guesses for each parameter at the same time and see what combinations of parameters values work best with each other for minimizing the error.

Despite the effort to optimize not only the parameters, but also the initial guesses we still arrived at the same conclusions. The code is available in Appendix VI. The optimized parameters and the optimized initial guesses we have used were again:

$$\alpha = 0.2774, \quad \beta = 0.0106, \quad \delta = 0.0375, \quad \gamma = 1.8381, \quad \lambda = 0.000754, \quad \kappa = 0.000144$$

and the minimized loss function value was:

$$\text{Final loss} = 142828.0842$$

We do not include the visualizations, since it would be the same as in Figure 5. Moreover, despite the effort, not only we did not further minimize the loss, but we were also not able to better fit the model to the actual data, as shown on Figure 5. Compared to the simple model shown on Figure 4, we made an improvement in the loss function value, despite the imperfect visual fit. We conclude that the inclusion of the damping coefficient improved the loss. Perhaps, more complex methods are necessary in order to fit both population simultaneously.

5.2 Markov Chain Monte Carlo

Another yet much more difficult and higher time complexity method for optimizing our prey-predator model can be using Bayesian Inference to find the best fitting parameters for our dataset. This method works with Markov Chain Monte Carlo (MCMC) to sample the posterior distribution of the parameters, and does the sampling process for a given number of warm up and actual sampling steps. After training the model, the code draws samples from the posterior using Predictive. And this gives us predicted population curves for hares and lynxes, along with uncertainty ranges (NumPyro, 2019). We include the below visualization and cite its source, as running the original code requires extensive time and resources.

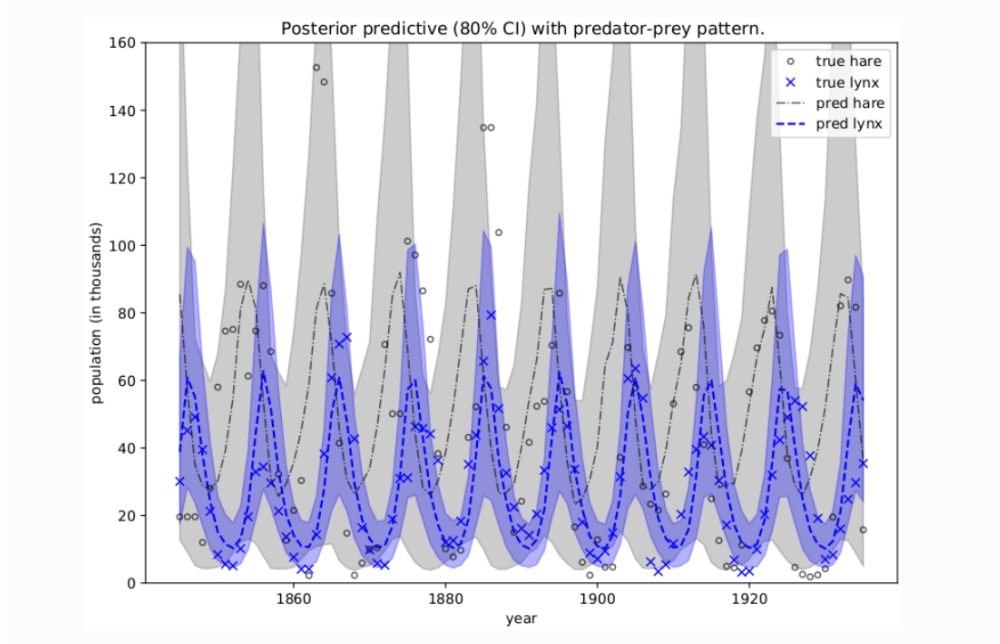


Figure 6: Fitted predator and prey model for Hares and Lynx using MCMC and Bayesian Inference (NumPyro, 2019)

As shown on Figure 6, the complex method seem to fit actual population data quite well. However, the predicted model still does not fit the hare population, similarly to our final findings. There may be many reasons for that, including external, environmental influences. Another problem may be related to the dataset itself, as it is acknowledged in the source that some data may be either missing, or be not fully verified (Hundley, 2003).

6 Results and Analysis

In this research project, we applied the Lotka-Volterra predator-prey model to historical lynx and hare population data in Canada to determine the best parameters.

6.1 Simple Predator and Prey Model

In the classical simple Prey-Predator Model without damping, we observed a constant oscillations with the population of the predator and prey. When the prey populations increase, the predator population of grew. On the other hand, when the predator population grew, over-predation will causes the prey number decline. This will lead to predator starvation and a drop in predator number.

In the simple damped Prey-Predator model, the prey and predator population still experiences oscillation. Nevertheless, the addition of the damping terms will cause the amplitude of the oscillation to decrease over time. The population of prey and predator will eventually settle at an equilibrium value where the population count is constant. This is similar to all things in natural which will eventually reach an equilibrium.

6.2 Inverse Problem - Undamped

Using the loss function defined early to conduct optimization, we discovered the following optimal parameter estimation yielded $\alpha = 1.3968$, $\beta = 0.0614$, $\delta = 0.0086$, and $\gamma = 0.2901$.

α is the Prey (hare) birth rate. An α of 1.3968 indicates rapid hare reproduction rate. Hares could quickly repopulate in the absence of predator.

β is the Prey (hare) Predation rate. $\beta = 0.0614$ reflects moderate predation pressure on Hares.

δ is the Predator(lynx) reproduction rate. $\delta = 0.0086$ indicates that Lynx gain very little energy from consuming each hare.

γ is the Predator(lynx) death rate. $\gamma = 0.2901$ means high lynx mortality(death) rate without food. This means Lynx could not survive long enough without food.

6.3 Inverse Problem - Damped

In the damped inverse problem, we got the optimal parameter of $\delta = 0.0086$ and $\gamma = 0.2901$.

λ is the limiting term of the prey. The Hare population is limited by food and space availability.

γ is the limiting term for the predator. The predator population is limited by lynx population via assumption on territoriality/disease.

The optimal parameters $\lambda = 0.000754$ and $\gamma = 0.000144$ implies that there is weak self-limitation effects on the population both Lynx and Hares. Nevertheless, the addition of these two parameter improved the lynx population fitting.

However, both models still struggled to perfectly predicted observed hare population extremes. This indicates potential missing mechanisms like environmental, alternative food sources, or influence from other species.

7 Conclusion

In this study, we utilized the Lotka-Volterra predator-prey model to analyze historical lynx and hare population data in Canada. To search for an optimal parameter, we minimize the sum of squared error computed from the loss function to arrive at the optimal parameters for the fitted model. The undamped fitted model captured a classic cyclic dynamics of a Lotka-Volterra predator-prey model. Nevertheless, it struggles to perfectly describe the pattern of the predator prey situation. Hence, we introduced the damped version, which incorporates self-limiting terms that provided more realistic stabilization of populations over time.

The parameter estimation revealed crucial and fascinating insights about the lynx-hare predator-prey system. For prey dynamics, the high α intrinsic growth rate ($\alpha = 1.40$) indicates hares can rapidly repopulate under favorable conditions. The moderate predation vulnerability ($\beta = 0.061$) of hares suggests that hare population faced constant but limited predation pressure. The low self-regulation term ($\lambda = 0.00075$) implies hare populations are largely limited by Lynx predation rather than competition for available food and living space.

For predator dynamics, the high mortality rate ($\gamma = 0.29-1.84$) highlights lynx vulnerability to starvation. This is also compounded by their inefficient energy abstraction from prey ($\delta = 0.0086$).

The minimal self-limitation ($\gamma = 0.00014$) further emphasizes that lynx populations are primarily controlled by hare availability rather than territorial limitation or social competition.

These parameters paint a picture of an interesting ecosystem where hares possess strong reproductive capacity but face constant predation pressure. The population of Lynx is highly dependent on hare available due to inefficiency at converting this resource into sustainable population growth.

The model's high residual fitting errors suggest there are unaccounted factors such as environmental variability, food sources, or other social factors which may be needed to fully explain the observed dynamics. In conclusion, these parameter estimates provide valuable information for understanding the Lotka-Volterra predator prey framework of Lynx and Hares in Canada.

In the future, we could incorporate more environmental factors such as climate and disease to better explain extreme population swings. Moreover, applying more complex methods along Bayesian inference to quantify parameter uncertainty could also improve model robustness.

References

- Alake, R. (2023). *Loss functions in machine learning explained* [Accessed: 2025-04-16]. <https://www.datacamp.com/tutorial/loss-function-in-machine-learning>
- Anisiu, M.-C. (2014). Lotka, volterra and their model. *Didactica Mathematica*, 32, 9–17.
- Errazki, M. (2022). *Simulating predator-prey models in python* [Accessed: 2025-04-16]. <https://medium.com/@errazkim/simulating-predator-prey-models-630f96607491>
- Hundley, D. (2003). Hare and lynx dataset 1845-1935 [Originaly based on Odum's "Fundamentals of Ecology" (p. 191), and MacLulich (1937). Retrieved from <http://people.whitman.edu/~hundledr/courses/M250F03/M250.html>].
- NumPyro. (2019). Example: Predator-prey model [Accessed: 2025-04-17]. <https://num.pyro.ai/en/0.7.2/examples/ode.html>
- Saint-Antoine, M. (2021). Parameter estimation - predator prey model (lotka-volterra) [Accessed: 2025-04-16]. https://github.com/mikesaint-antoine/Comp_Bio_Tutorials/blob/main/parameter_estimation/predator_prey/fit_model.py
- Wikipedia contributors. (2020). Weighted least squares [Accessed: 2025-04-16].

Appendices

I Appendix

```
[ ]: import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Lotka-Volterra
def lotka_volterra(z, t, alpha, beta, gamma, delta):
    x, y = z
    dxdt = alpha * x - beta * x * y
    dydt = -gamma * y + delta * x * y
    return [dxdt, dydt]

# Time
t = np.linspace(0, 15, 500)

# Initial population
z0 = [1, 0.5]

# Parameters from (Anisiu, 2014)
alpha = 2    # prey birth rate
beta = 1.1   # predation rate
gamma = 1    # predator death rate
delta = 0.9  # predator reproduction rate

# SOLVE
sol = odeint(lotka_volterra, z0, t, args=(alpha, beta, gamma, delta))

# PLOT
plt.figure(figsize=(7, 5))
plt.plot(t, sol[:, 0], label='Prey population (x)', color='tab:blue')
plt.plot(t, sol[:, 1], label='Predator population (y)', color='tab:red')

plt.xlabel('Time (t)')
plt.xlim(0, 15)
plt.ylabel('Population (x and y)')
plt.legend(loc='upper right')

plt.grid(True)
plt.tight_layout()

plt.savefig('simulated1.png')

plt.show()
```

II Appendix

(Anisiu, 2014)

```
[ ]: import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Lotka-Volterra
def lotka_volterra(z, t, alpha, beta, gamma, delta):
    x, y = z
    dxdt = alpha * x - lam * x * x - beta * x * y
    dydt = -gamma * y - kappa * y * y + delta * x * y
    return [dxdt, dydt]

# Time
t = np.linspace(0, 15, 500)

# Initial population
z0 = [1, 0.5]

# Parameters from (Anisiu, 2014)
alpha = 2    # prey birth rate
beta = 1.1   # predation rate
gamma = 1    # predator death rate
delta = 0.9  # predator reproduction rate
lam = 0.1
kappa = 0.1

# SOLVE
sol = odeint(lotka_volterra, z0, t, args=(alpha, beta, gamma, delta))

# PLOT
plt.figure(figsize=(7, 5))
plt.plot(t, sol[:, 0], label='Prey population (x)', color='tab:blue')
plt.plot(t, sol[:, 1], label='Predator population (y)', color='tab:red')

plt.xlabel('Time (t)')
plt.xlim(0, 15)
plt.ylabel('Population (x and y)')
plt.legend(loc='upper right')

plt.grid(True)
plt.tight_layout()

plt.savefig('simulated2.png')

plt.show()
```

III Appendix

```
[ ]: import matplotlib.pyplot as plt
import numpy as np

raw_data = """(http://people.whitman.edu/~hundledr/courses/M250F03/LynxHare.
↳txt)"""

years, hares, lynxes = [], [], []
for line in raw_data.strip().split("\n"):
    y, h, l = line.split()
    years.append(float(y))
    hares.append(float(h))
    lynxes.append(float(l))

# Observed Data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6), sharex=True)

ax1.plot(years, hares, color='blue', label="Observed Hares")
ax1.set_title("Hare Population", fontsize=16)
ax1.set_ylabel("Hares", fontsize=16)
ax1.set_xlabel("Year", fontsize=16)
ax1.tick_params(axis='both', labels=16)
ax1.grid(True)
ax1.legend(fontsize=15)

ax2.plot(years, lynxes, color='red', label="Observed Lynx")
ax2.set_title("Lynx Population", fontsize=16)
ax2.set_ylabel("Lynx", fontsize=16)
ax2.set_xlabel("Year", fontsize=16)
ax2.tick_params(axis='both', labels=16)
ax2.grid(True)
ax2.legend(fontsize=15)

xticks = np.arange(min(years), max(years)+1, 10)
ax1.set_xticks(xticks)
ax2.set_xticks(xticks)

plt.tight_layout()
plt.savefig("realdata.png")
plt.show()
```

IV Appendix

(Saint-Antoine, 2021)

```
[ ]: import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize
from scipy.integrate import odeint

# Lotka-Volterra
def sim(variables, t, params):
    x, y = variables # hares, lynx
    alpha, beta, delta, gamma = params
    dxdt = alpha * x - beta * x * y
    dydt = delta * x * y - gamma * y
    return [dxdt, dydt]

# Loss Function
def loss_function(params, years, hares, lynxes):
    y0 = [hares[0], lynxes[0]]
    t = np.linspace(years[0], years[-1], num=len(years))
    output = odeint(sim, y0, t, args=(params,))
    loss = np.sum((output[:,0] - hares)**2 + 2*(output[:,1] - lynxes)**2)
    return loss

# Parameter fitting
params0 = np.array([0.6, 0.02, 0.1, 0.8])
best_params, final_loss, iters, funcalls, warnflag = scipy.optimize.
    ↪fmin(loss_function, params0, args=(years, hares, lynxes),
    ↪disp=True,full_output=True)

alpha_fit, beta_fit, delta_fit, gamma_fit = best_params

# Simulations
params = [alpha_fit, beta_fit, delta_fit, gamma_fit]
y0 = [hares[0], lynxes[0]]
t_fine = np.linspace(years[0], years[-1], num=1000)
output = odeint(sim, y0, t_fine, args=(params,))

# PLOT
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6), sharex=True)

ax1.plot(years, hares, color='blue', label="Observed Hares")
ax1.plot(t_fine, output[:, 0], 'b--', label="Fitted Hares")
ax1.set_title("Hare Population", fontsize=16)
ax1.set_ylabel("Hares", fontsize=16)
ax1.set_xlabel("Year", fontsize=16)
```

```

ax1.tick_params(axis='both', labels=15)
ax1.grid(True)
ax1.legend(fontsize=16)

ax2.plot(years, lynxes, color='red', label="Observed Lynx")
ax2.plot(t_fine, output[:, 1], 'r--', label="Fitted Lynx")
ax2.set_title("Lynx Population", fontsize=16)
ax2.set_ylabel("Lynx", fontsize=16)
ax2.set_xlabel("Year", fontsize=16)
ax2.tick_params(axis='both', labels=15)
ax2.grid(True)
ax2.legend(fontsize=16)

xticks = np.arange(int(min(years)), int(max(years)) + 1, 10)
ax1.set_xticks(xticks)
ax2.set_xticks(xticks)

plt.tight_layout()
plt.savefig("fitted1.png")
plt.show()

# Fitted parameters
print("Fitted Parameters:")
print(f"alpha = {alpha_fit:.4f}")
print(f"beta  = {beta_fit:.4f}")
print(f"delta = {delta_fit:.4f}")
print(f"gamma = {gamma_fit:.4f}")
print(f"Final minimized loss: {final_loss:.4f}")

```

V Appendix

```
[ ]: import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize
from scipy.integrate import odeint

# Lotka-Volterra with damping
def sim(variables, t, params):
    x, y = variables # hares, lynx
    alpha, beta, delta, gamma, lambd, kappa = params
    dxdt = alpha * x - beta * x * y - lambd * x**2
    dydt = delta * x * y - gamma * y - kappa * y**2
    return [dxdt, dydt]

# Loss Function
def loss_function(params, years, hares, lynxes):
    y0 = [hares[0], lynxes[0]]
    t = np.linspace(years[0], years[-1], num=len(years))
    output = odeint(sim, y0, t, args=(params,))
    loss = np.sum((output[:,0] - hares)**2 + 2 * (output[:,1] - lynxes)**2)
    return loss

# Parameter fitting
params0 = np.array([0.8, 0.03, 0.04, 0.4, 0.0003, 0.0003]) #  $\alpha$ ,  $\beta$ ,  $\delta$ ,  $\gamma$ ,  $\lambda$ ,  $\kappa$ 
best_params, final_loss, iters, funcalls, warnflag = scipy.optimize.
    ↳fmin(loss_function, params0, args=(years, hares, lynxes),
    ↳disp=True, full_output=True)

alpha_fit, beta_fit, delta_fit, gamma_fit, lambd_fit, kappa_fit = best_params

# Simulations
params = [alpha_fit, beta_fit, delta_fit, gamma_fit, lambd_fit, kappa_fit]
y0 = [hares[0], lynxes[0]]
t_fine = np.linspace(years[0], years[-1], num=1000)
output = odeint(sim, y0, t_fine, args=(params,))

# PLOT
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6), sharex=True)

ax1.plot(years, hares, color='blue', label="Observed Hares")
ax1.plot(t_fine, output[:, 0], 'b--', label="Fitted Hares")
ax1.set_title("Hare Population with Damping", fontsize=16)
ax1.set_ylabel("Hares", fontsize=16)
ax1.set_xlabel("Year", fontsize=16)
```

```

ax1.tick_params(axis='both', labels=15)
ax1.grid(True)
ax1.legend(fontsize=16)

ax2.plot(years, lynxes, color='red', label="Observed Lynx")
ax2.plot(t_fine, output[:, 1], 'r--', label="Fitted Lynx")
ax2.set_title("Lynx Population with Damping", fontsize=16)
ax2.set_ylabel("Lynx", fontsize=16)
ax2.set_xlabel("Year", fontsize=16)
ax2.tick_params(axis='both', labels=15)
ax2.grid(True)
ax2.legend(fontsize=16)

xticks = np.arange(int(min(years)), int(max(years)) + 1, 10)
ax1.set_xticks(xticks)
ax2.set_xticks(xticks)

plt.tight_layout()
plt.savefig("fitted_damped.png")
plt.show()

# Fitted parameters
print("Fitted Parameters (with Damping):")
print(f"alpha = {alpha_fit:.4f}")
print(f"beta = {beta_fit:.4f}")
print(f"delta = {delta_fit:.4f}")
print(f"gamma = {gamma_fit:.4f}")
print(f"lambda = {lambda_fit:.6f}")
print(f"kappa = {kappa_fit:.6f}")
print(f"Final minimized loss: {final_loss:.2f}")

```

VI Appendix

```
[ ]: import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize
from scipy.integrate import odeint

#Lotka-Volterra with damping
def sim(variables, t, params):
    x, y = variables # hares, lynx
    alpha, beta, delta, gamma, lambd, kappa = params
    dxdt = alpha * x - beta * x * y - lambd * x**2
    dydt = delta * x * y - gamma * y - kappa * y**2
    return [dxdt, dydt]

#Loss Function
def loss_function(params, years, hares, lynxes):
    y0 = [hares[0], lynxes[0]]
    t = np.linspace(years[0], years[-1], num=len(years))
    output = odeint(sim, y0, t, args=(params,))
    loss = np.sum((output[:,0] - hares)**2 + 2 * (output[:,1] - lynxes)**2)
    return loss

# Various initial guesses for all parameters to make more accurate optimization
↪ a, β, δ, γ, λ, κ

initial_guesses = [
    [1, 1, 1, 1, 1, 1],
    [0.5, 0.05, 0.05, 0.5, 0.00031, 0.0001],
    [0.8, 0.0033, 0.04, 0.4, 0.0003, 0.0003],
    [0.8, 0.03, 0.04, 0.4, 0.0003, 0.0003], # the value from Appendix V
    [0.8, 0.025, 0.03999, 0.399, 0.0003, 0.0003],
    [0.78, 0.039, 0.039, 0.39, 0.00039, 0.00039],
    [0.8, 0.0399, 0.0389, 0.389, 0.000389, 0.000389]
]

all_losses = []
all_params = []

for params0 in initial_guesses:
    result = scipy.optimize.fmin(loss_function, params0, args=(years, hares, ↪
    ↪ lynxes), disp=False, full_output=True)
    best_params, final_loss = result[0], result[1]
    all_params.append(best_params)
```

```

all_losses.append(final_loss)

min_idx = np.argmin(all_losses)

best_params = all_params[min_idx]
final_loss = all_losses[min_idx]
best_initials = initial_guesses[min_idx]

alpha_fit, beta_fit, delta_fit, gamma_fit, lambd_fit, kappa_fit = best_params

params = [alpha_fit, beta_fit, delta_fit, gamma_fit, lambd_fit, kappa_fit]
y0 = [hares[0], lynxes[0]]
t_fine = np.linspace(years[0], years[-1], num=1000)
output = odeint(sim, y0, t_fine, args=(params,))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6), sharex=True)

# Hares
ax1.plot(years, hares, color='blue', label="Observed Hares")
ax1.plot(t_fine, output[:, 0], 'b--', label="Fitted Hares")
ax1.set_title("Hare Population with Damping", fontsize=16)
ax1.set_ylabel("Hares", fontsize=16)
ax1.set_xlabel("Year", fontsize=16)
ax1.tick_params(axis='both', labels=15)
ax1.grid(True)
ax1.legend(fontsize=16)

# Lynxes
ax2.plot(years, lynxes, color='red', label="Observed Lynx")
ax2.plot(t_fine, output[:, 1], 'r--', label="Fitted Lynx")
ax2.set_title("Lynx Population with Damping", fontsize=16)
ax2.set_ylabel("Lynx", fontsize=16)
ax2.set_xlabel("Year", fontsize=16)
ax2.tick_params(axis='both', labels=15)
ax2.grid(True)
ax2.legend(fontsize=16)

xticks = np.arange(int(min(years)), int(max(years)) + 1, 10)
ax1.set_xticks(xticks)
ax2.set_xticks(xticks)

plt.tight_layout()
plt.savefig("fitted_damped_2.png")
plt.show()

```

```
print("Fitted Parameters (with Damping):")
print(f"alpha  = {alpha_fit:.4f}")
print(f"beta   = {beta_fit:.4f}")
print(f"delta  = {delta_fit:.4f}")
print(f"gamma  = {gamma_fit:.4f}")
print(f"lambda = {lambda_fit:.6f}")
print(f"kappa  = {kappa_fit:.6f}")
print(f"Final minimized loss: {final_loss:.4f}")
print("Best initial guesses:", best_initials)
```