Lotka-Volterra Model

Population Dynamics of Rabbits and Foxes

AMS 333

Mathematical Biology

Homework 3

Wenhan Gao

State University of New York at Stony Brook

November 2020

1 Introduction

The study of population dynamics is about how and why populations change in size and structure over time. Different species reproduce and interact in different ways, and we have different mathematical models for them. In this report, we will focus on the population dynamics of rabbits and foxes using the Lotka-Volterra Model. The Lotka-Volterra model for prey-predator relationships is one of the most fundamental models in population dynamics.Also, in this report, Matlab, a computer programming language and tool, will be utilized to simulate the dynamical system, and create plots to show trends help us analyze and understand such systems.

2 Mathematical Modeling

2.1 Simple Lotka-Volterra Model

In this report, the prey population are the rabbits, and predators are the foxes. The simple Lotka-Volterra model is described by two differential equations indicating predator and prey population changes.

$$\frac{dN_{prey}}{dt} = +R_{prey,0}N_{prey}(t) - \gamma N_{prey}(t)N_{pred.}(t) \tag{2.1}$$

$$\frac{dN_{pred.}}{dt} = \epsilon\gamma N_{prey}(t)N_{pred.}(t) - R_{pred.,0}N_{pred.}(t) \tag{2.2}$$

$N(t)$ represents the population size per km^2 at a given time t, $R_{prey,0} = 0.04$ is the per capita natural growth rate of rabbits in the absence of predation, and we can deduce that the **doubling time of rabbits without predation** is $\ln 2/R_{prey,0} = 17.33$ days. $R_{pred.,0} = 0.2$ is the **natural death rate of foxes in the absence of rabbits**. **These two parameters are biologically reasonable** because each breeding cycle of rabbits will take 65 to 75 days, and each time 4 to 12 babies are born(remember that only female rabbits can reproduce). The raw death rate of foxes is 0.2, so let's consider the simple exponential decay model for foxes in the absence of food:$N_{prey}(t) = N_{prey,0}e^{-0.2t}$, and $e^{-0.2\cdot 5} = 0.37$, which means after 5 days without food, the foxes population size will shrink more than half, and $e^{-0.2\cdot 15} = 0.05$, which means, after 15 days without food, there are barely any alive fox.

$\gamma = 0.0005$ is the **predation constant**, and $\epsilon = 0.1$ is **a constant that describes therelationship between the preadtion rate and predator growth**. The two parameters **do make sense**. The non-zero stationary

point of this system is when $N_{prey} = \frac{R_{pred.,0}}{\epsilon\gamma} = \frac{0.2}{0.1\times0.0005} = 4000$ and $N_{pred.} = \frac{R_{prey.,0}}{\gamma} = \frac{0.04}{0.0005} = 80$, and 4000 rabbits and 80 foxes per km^2 do make physical sense.

Now, let's implement simple Lotka-Volterra model in Matlab.

Figure 1: Simple Lotka-Volterra code part 1

```
Nprey = zeros(36500,1);
Npred = zeros(36500,1);
dNprey_dt = zeros(36500,1);
dNpred_dt = zeros(36500,1);
% make vectors of size (36500, 1)
Nprey(1) = 200;
Npred(1) = 50;
% initial value
% change to 5000, 1000 and 4000,80 later on
deltaTau = 0.01;
% step of 0.01 day
Rprey = 0.04;
Rpred = 0.2;
gamma = 0.0005;
epsilon = 0.1;
% input parameters
for i = 1:36500
    dNprey_dt(i) = (Rprey * Nprey(i)) - gamma * Nprey(i) * Npred(i);
    dNpred_dt(i) = (epsilon * gamma * Nprey(i) * Npred(i)) - Rpred *
Npred(i);
    Nprey(i + 1) = Nprey(i) + (dNprey_dt(i) * deltaTau);
    Npred(i + 1) = Npred(i) + (dNpred_dt(i) * deltaTau);
end
% applying forward Euler's method
t = 0:0.01:365;
plot(t,Nprey/30,'g');
hold on
plot(t,Npred,'r');
xlabel('Time in days');
ylabel('Population');
title('Initial 200 rabbits and 50 foxes');
axis([1 365 0 Inf]);
legend('Prey/30','Predator');
hold off
prey_range = [min(Nprey), max(Nprey)]
pred_range = [min(Npred), max(Npred)]
```

Above code part 1 yields plot of population versus time, the following part 2 is a continuation of above code, and yields plot of population versus each other with velocity arrows.

Figure 2: Simple Lotka-Volterra code part 2

```
plot(Nprey,Npred);
xlabel('Number of prey');
ylabel('Number of predator');
hold on;
% the following code creates the velocity field
max_U = max(Nprey);
max_V = max(Npred);
N = 20;
range_U = 0:(max_U/N):max_U;
range_V = linspace(0,max_V,N+1);
[UU,VV] = meshgrid(range_U, range_V);
vel_U = alpha*UU - gamma*UU.*VV;
vel_V = epsilon*gamma*UU.*VV - beta*VV;
h = quiver(range_U,range_V,vel_U,vel_V,0.5);
set (h, "maxheadsize", 0.005);
axis([0 Inf 0 Inf]);
hold off;
```

End of Figure 2

By initial populations of 200 rabbits and 50 foxes, the Matlab results are:

Figure 3: Population Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^4 ×
    0.0189    1.8483
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
10^3 ×
    0.0000    1.1297
```

Figure 4: Population Versus Time

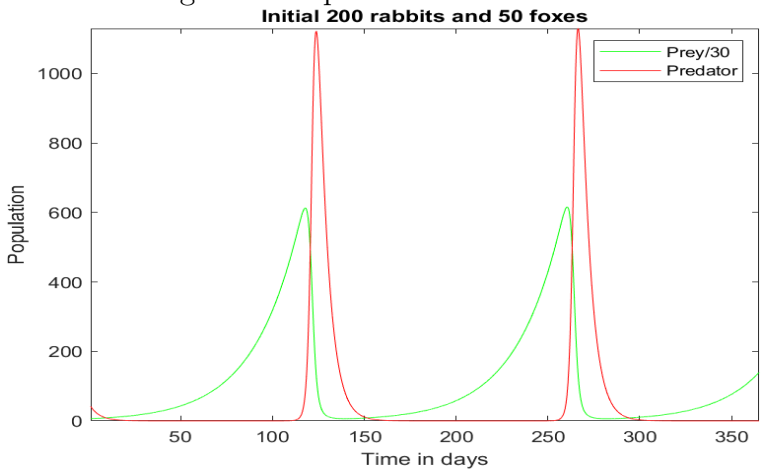
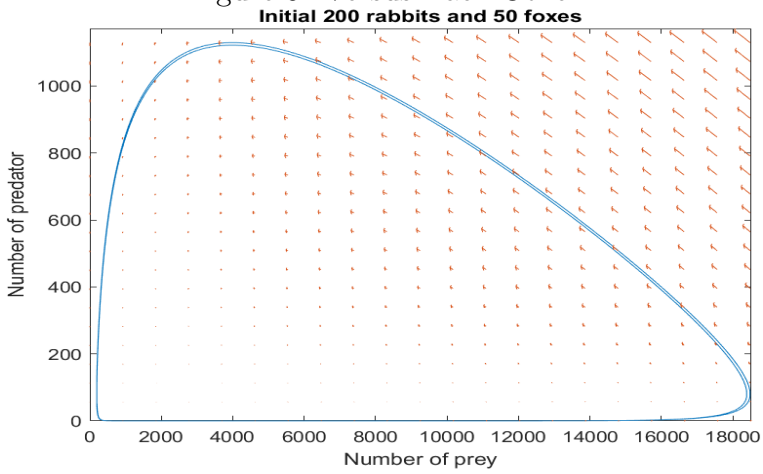


Figure 5: Versus Each Other



As in the above plots, we can see periodic motion of both rabbits/prey and foxes/predators. **The prey population ranges from 189 to 18483 and the predator population ranges from 0 to around 1130.** When predator population is low, prey population grows almost exponentially, and as prey population grows to a high level, predator population grows almost exponentially as well. Moreover, as predator population gets larger, prey population decreases.The predator also decreases as prey population decreases and can die out eventually when the prey population be is too low to support the predator population.

Next, by initial populations of 5000 rabbits and 100 foxes, a similar code as above produces the following results:

Figure 6: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    3.0578    5.1200
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
   42.3464  135.1495
```

Figure 7: Population Versus Time

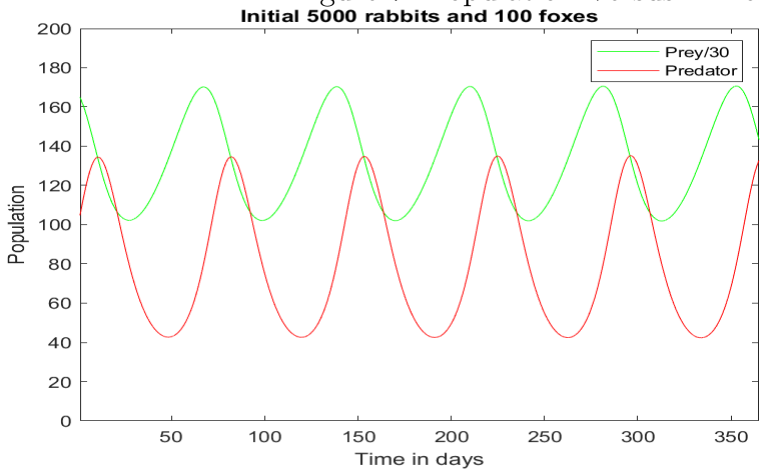
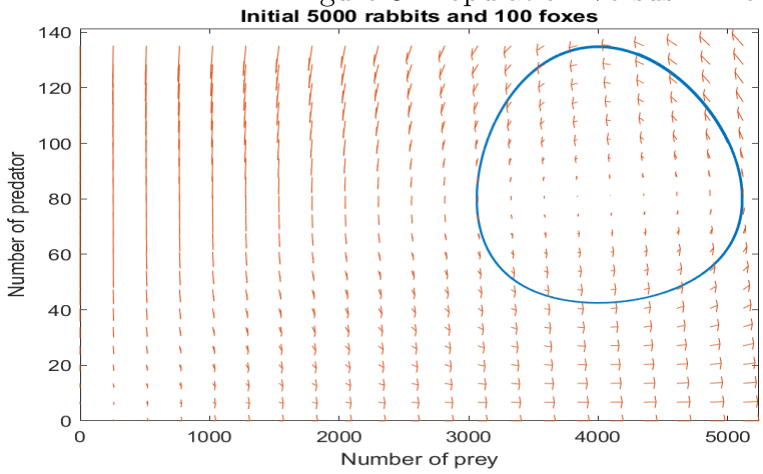


Figure 8: Population Versus Time



In this case, we still observe periodic motion. The main difference between this result and previous result is that the predator population never die out. This is because the prey population is always high enough to prevent predator population from ever reaching zero. This shows that different sets of starting conditions gives different periodic solution and the system does not have a stable limit cycle.

Finally, by initial populations of 4000 rabbits and 80 foxes, a similar code as above produces the following results:

Figure 9: Population Versus Time

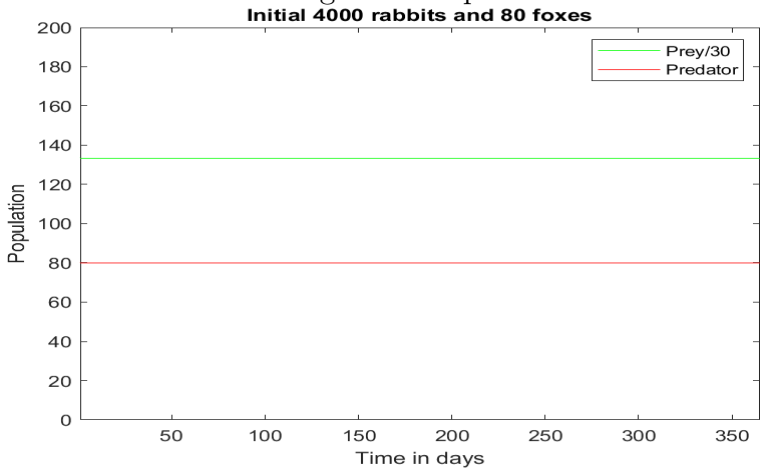


Figure 10: Population Versus Time

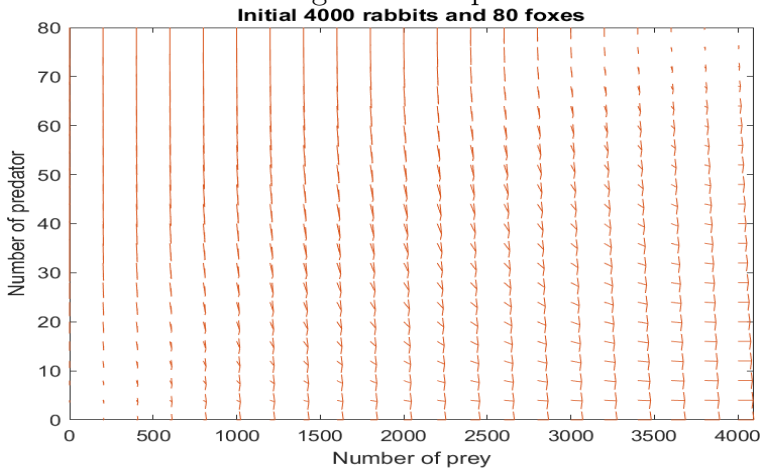


Figure 11: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
        4000        4000
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
     80    80
```

In this case, we no longer observe periodic motion. Instead, we observe that both prey and predator populations stay constant. This is expected because any system beginning at a stationary point will not evolve over time. This is a state when the prey and predator populations have equilibrium. This indicates a difference from the previous two results in a way that population size does not vary over time.

2.2 Extending the Lotka–Volterra Model

2.2.1 Logistic Based Growth

In the simple Lotka-Volterra Model, we assume that prey population grows exponentially in the absence of the predators. However, this is not ideal for realistic situations. To make the simulation more accurate, we use logistic equation based model to restrict the growth of prey population in the absence of predators by replacing R_{prey} with $R_{prey}(1 - \frac{N_{prey}}{K})$, where K is the carrying capacity of rabbits representing maximal prey population that the environment can support, and it's said to be 10,000. When prey population approaches K, the growth rate R approaches 0. Matlab code:

Figure 12: Lotka-Volterra with Carrying Capacity Matlab Code

```
Nprey = zeros(36500*3,1); Npred = zeros(36500*3,1);
dNprey_dt = zeros(36500*3,1); dNpred_dt = zeros(36500*3,1);
% make vectors of size (36500*3, 1)
Nprey(1) = 200;
Npred(1) = 50;
% initial value
% change to 5000, 1000 and 4000,80 later on
deltaTau = 0.01;
% step of 0.01 day
Rprey = 0.04; Rpred = 0.2;
gamma = 0.0005; epsilon = 0.1;
s = gamma; K = 10000; h = 0.2;
% input parameters
```

Continuation of Figure 12

```
for i = 1:36500*3
    dNprey_dt(i) = Rprey * Nprey(i)*(1-Nprey(i)/K) - gamma * Nprey(i) *
Npred(i);
    dNpred_dt(i) = (epsilon * gamma * Nprey(i) * Npred(i)) - Rpred *
Npred(i);
    Nprey(i + 1) = Nprey(i) + (dNprey_dt(i) * deltaTau);
    Npred(i + 1) = Npred(i) + (dNpred_dt(i) * deltaTau);
end
% applying forward Euler's method
t = 0:0.01:365*3;
plot(t,Nprey/100,'g');
hold on
plot(t,Npred,'r');
xlabel('Time in days');
ylabel('Population');
title('Initial 200 rabbits and 50 foxes');
axis([1 365*3 0 Inf]);
legend('Prey/100','Predator');
hold off
prey_range = [min(Nprey), max(Nprey)]
pred_range = [min(Npred), max(Npred)]
plot(Nprey,Npred);
title('Initial 200 rabbits and 50 foxes');
xlabel('Number of prey');
ylabel('Number of predator');
hold on;
max_U = max(Nprey);
max_V = max(Npred);
N = 20;
range_U = 0:(max_U/N):max_U;
range_V = linspace(0,max_V,N+1);
[UU,VV] = meshgrid(range_U, range_V);
vel_U = Rprey*UU - gamma*UU.*VV;
vel_V = epsilon*gamma*UU.*VV - Rpred*VV;
h = quiver(range_U,range_V,vel_U,vel_V,0.5);
set (h, "maxheadsize", 0.005);
axis([0 Inf 0 Inf]);
hold off;
```

Firstly, the results with initial 200 rabbits and 50 foxes are:

Figure 13: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    0.2000    8.9762
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
    0.0003  232.6598
```

Figure 14: Population Versus Time

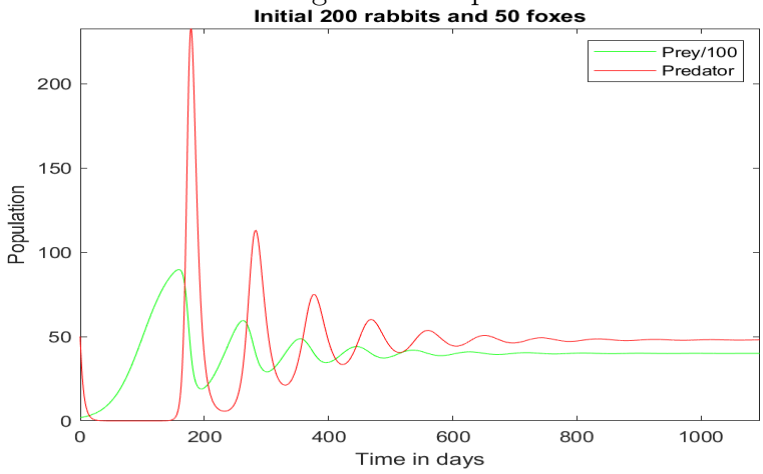
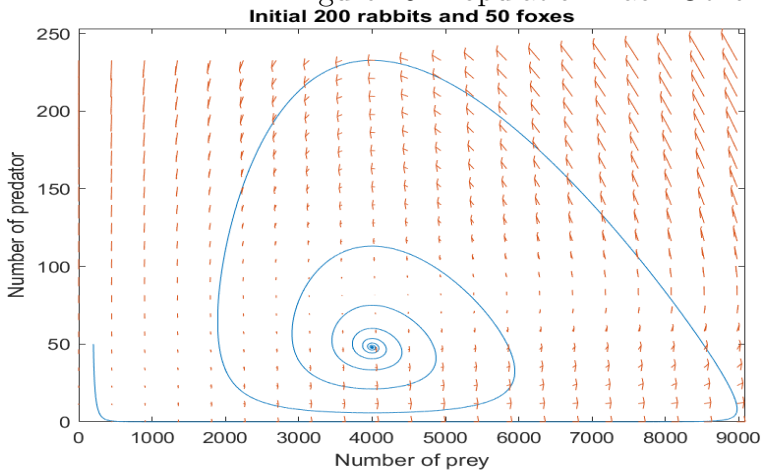


Figure 15: Population Each Other



We can see from above results that, with an initial population of 200 rabbits and 50 foxes, the prey population ranges from 200 to around 9000 rabbits, and the predator population ranges from 0 to around 233 foxes. Different from previous model, we do not observe periodic motion anymore. At low prey population, prey population increases while predator population decreases. When prey population is high, predator population increases rapidly, and prey population starts to shrink. This process, whose plot looks like wave curves, will repeat a few times, and the wave height gets smaller and smaller. Eventually, both prey and predator population stabilize at 4000 rabbits and 50 foxes. This model is different from the previous model because we introduced carrying capacity of prey population.

Next, results with initial 5000 rabbits and 100 foxes are:

Figure 16: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    3.0744    4.7225
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
   24.6646  100.0000
```

Figure 17: Population Versus Time

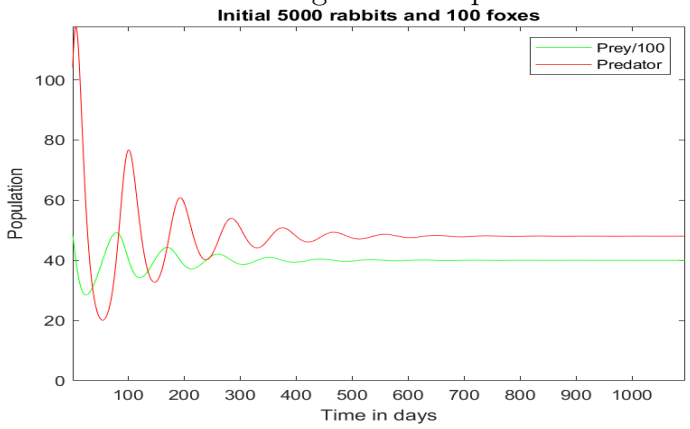
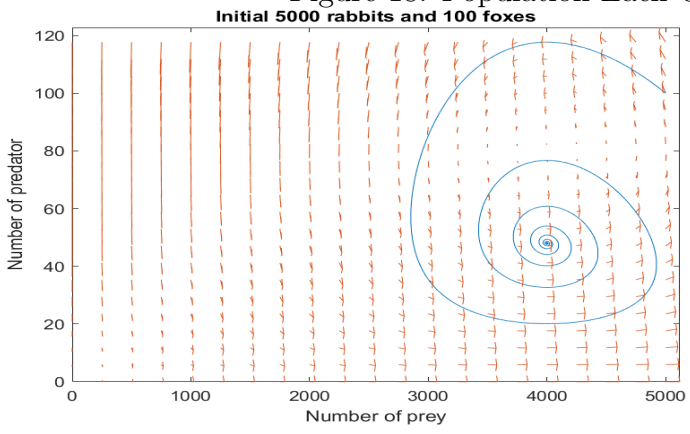


Figure 18: Population Each Other



We have similar result as we start with 200 rabbits and 50 foxes, but differences between maximal size and minimal size are smaller.The system also stabilizes at around 4000 rabbits and 50 foxes.

Finally, results with initial 4000 rabbits and 80 foxes are:

Figure 19: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    3.3692    4.4738
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
   31.3913   80.0000
```

Figure 20: Population Versus Time

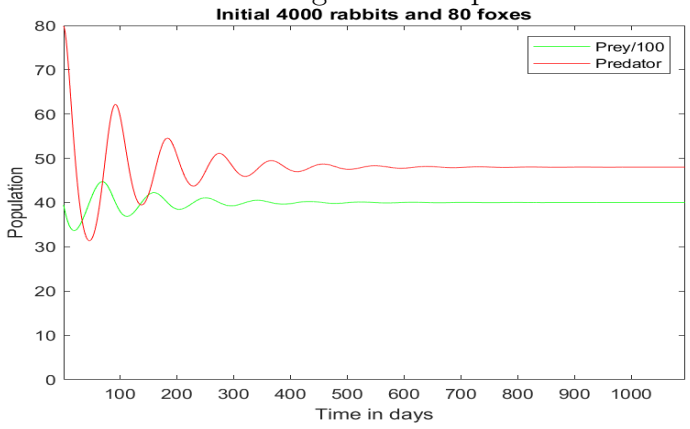
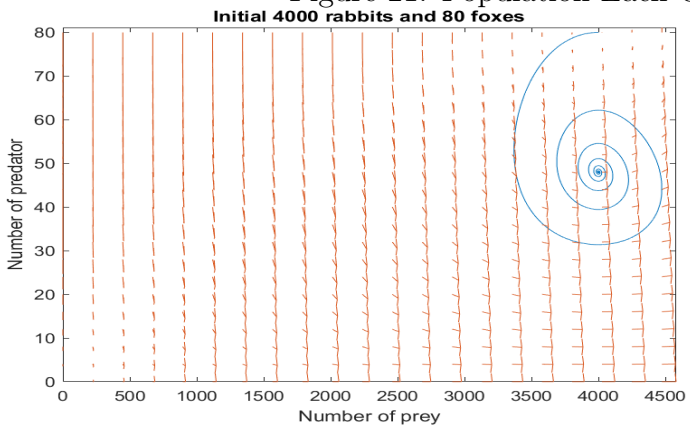


Figure 21: Population Each Other



Again, we have similar results. The prey population ranges from around 3369 to around 4473 rabbits. The predator population ranges from 32 to 80

foxes. Differences are even smaller than the one starting with 5000 rabbits and 100 foxes. The system stabilizes at around 4000 rabbits and 48 foxes.

By comparing results with and without carrying capacity, we observe that the system with carrying capacity will stabilize eventually; the system without carrying capacity will result in periodic motion except when the initial data set is the stationary point, 4000 rabbits and 80 foxes. The reason for this difference is: The realistic limitation restricts the speed of rabbit growth, so the amount of rabbits can only support less foxes. In the model with carrying capacity, I expect (4000 rabbits, 50 foxes) to be a stationary point since results with different initial values all stabilizes around this point, and this point is stable since velocity lines are moving counterclockwise towards this point. Also, I expect other two stationary points: (0 rabbits, 0 foxes), and (10000 rabbits, 0 fox), and both of them are stable stationary points for obvious reasons.

2.2.2 LV Model with Carrying Capacity and Predation Limit

It is not enough to just introduce carrying capacity. We also need to consider how an individual predator spends its time to haunt:(1)Searching for prey; (2)Making kills; (3)Eating; (4)Rests until hungry again. Therefore, we need to also introduce the predator response equation to take the time predator spend into account by replacing γN_{prey} with $\frac{sN_{prey}}{1+shN_{prey}}$, where s is proportionality constant and h is constant time consumed for every kill. When prey become abundant, $\frac{sN_{prey}}{1+shN_{prey}}$ becomes $\frac{1}{h}$, the search time approaches zero, but the time for killing, eating, and resting remains constant. Now let's use following code to implement this model. Note that the only part that has changed is the for-loop part, where we replaced γ.

Figure 22: Lotka-Volterra with Carrying Capacity Matlab Code

```
Nprey = zeros(36500*3,1); Npred = zeros(36500*3,1);
dNprey_dt = zeros(36500*3,1); dNpred_dt = zeros(36500*3,1);
% make vectors of size (36500*3, 1)
Nprey(1) = 200;
Npred(1) = 50;
% initial value
% change to 5000, 1000 and 4000,80 later on
deltaTau = 0.01;
% step of 0.01 day
Rprey = 0.04; Rpred = 0.2;
gamma = 0.0005; epsilon = 0.1;
s = gamma; K = 10000; h = 0.2;
% input parameters
```

Continuation of Figure 22

```
for i = 1:36500*3
    dNprey_dt(i) = Rprey * Nprey(i)*(1-Nprey(i)/K) -
(s*Nprey(i)/(1+s*h*Nprey(i))) * Npred(i);
    dNpred_dt(i) = (epsilon * (s*Nprey(i)/(1+s*h*Nprey(i))) * Npred(i)) -
Rpred * Npred(i);
    Nprey(i + 1) = Nprey(i) + (dNprey_dt(i) * deltaTau);
    Npred(i + 1) = Npred(i) + (dNpred_dt(i) * deltaTau);
end
% applying forward Euler's method
t = 0:0.01:365*3;
plot(t,Nprey/100,'g');
hold on
plot(t,Npred,'r');
xlabel('Time in days');
ylabel('Population');
title('Initial 200 rabbits and 50 foxes');
axis([1 365*3 0 Inf]);
legend('Prey/100','Predator');
hold off
prey_range = [min(Nprey), max(Nprey)]
pred_range = [min(Npred), max(Npred)]
fixedpt = [Nprey(end), Npred(end)]
plot(Nprey,Npred);
title('Initial 200 rabbits and 50 foxes');
xlabel('Number of prey');
ylabel('Number of predator');
hold on;
max_U = max(Nprey);
max_V = max(Npred);
N = 20;
range_U = 0:(max_U/N):max_U;
range_V = linspace(0,max_V,N+1);
[UU,VV] = meshgrid(range_U, range_V);
vel_U = Rprey*UU - gamma*UU.*VV;
vel_V = epsilon*gamma*UU.*VV - Rpred*VV;
h = quiver(range_U,range_V,vel_U,vel_V,0.5);
set (h, "maxheadsize", 0.005);
axis([0 Inf 0 Inf]);
hold off;
```

With an initial of 200 rabbits and 50 foxes, the results are:

Figure 23: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    0.2000    9.9933
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
    0.0001   72.5513
```

```
fixedpt = [Nprey(end), Npred(end)]
```

```
fixedpt = 1×2
10^3 ×
    6.6673    0.0445
```

Figure 24: Population Versus Time

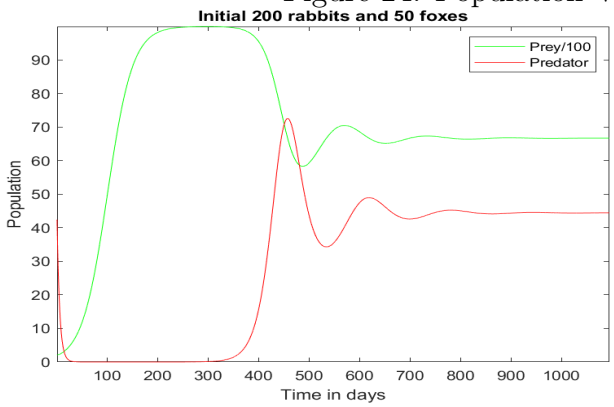
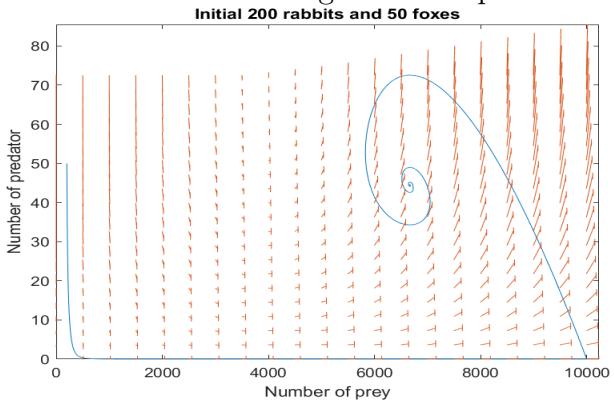


Figure 25: Population Each Other



With an initial of 200 rabbits and 50 foxes, values of the prey population range from around 200 to around 10000 rabbits, and of the predator population range from 0 to around 70 foxes. When the prey population is low, prey increases almost exponentially while predator population decreases because prey population is not high enough to support them. When prey population is high, predator population starts to grow, and as a consequence, prey population starts to shrink. Eventually, the system stabilizes at around 6667 rabbits and 45(rounded up from 44.5) foxes.

Next, with an initial of 5000 rabbits and 100 foxes, the results are:

Figure 26: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    4.6367    7.6837
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
   20.5501  100.0000
```

```
fixedpt = [Nprey(end), Npred(end)]
```

```
fixedpt = 1×2
10^3 ×
    6.6667    0.0444
```

Figure 27: Population Versus Time

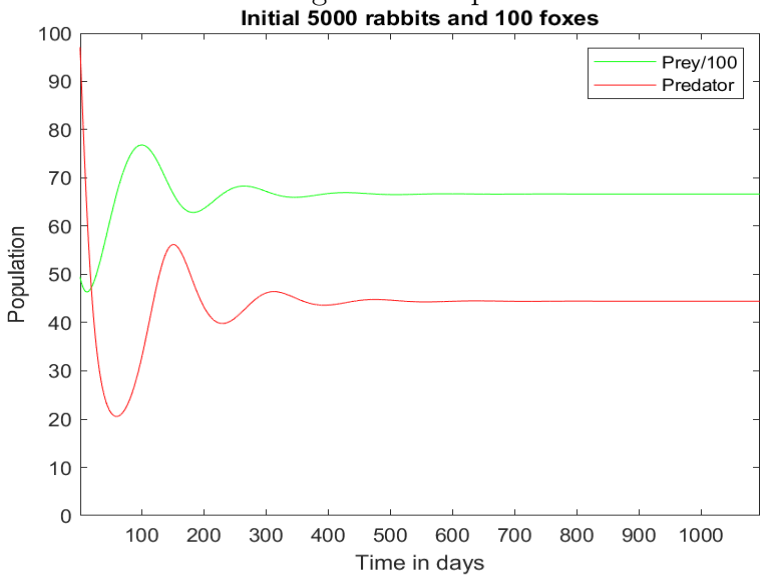
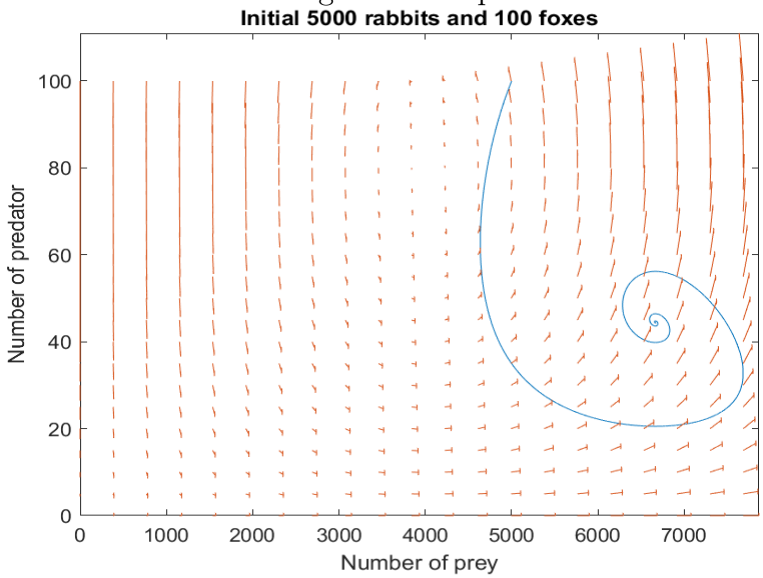


Figure 28: Population Each Other



With an initial of 5000 rabbits and 100 foxes, the prey population ranges from around 4637 to 7684 rabbits, and the predator population ranges from around 21 to 100 foxes.The system behaves similar to the one with an initial of 200 rabbits and 50 foxes, and the system stabilizes at 6667 rabbits(rounded up from 6666.7) and 45 foxes(rounded up from 44.4).
Lastly, with an initial of 4000 rabbits and 80 foxes, the results are:

Figure 29: Range

```
prey_range = [min(Nprey), max(Nprey)]
```

```
prey_range = 1×2
10^3 ×
    3.9735    8.0787
```

```
pred_range = [min(Npred), max(Npred)]
```

```
pred_range = 1×2
   13.8849   80.0000
```

```
fixedpt = [Nprey(end), Npred(end)]
```

```
fixedpt = 1×2
10^3 ×
    6.6667    0.0444
```

Figure 30: Population Versus Time

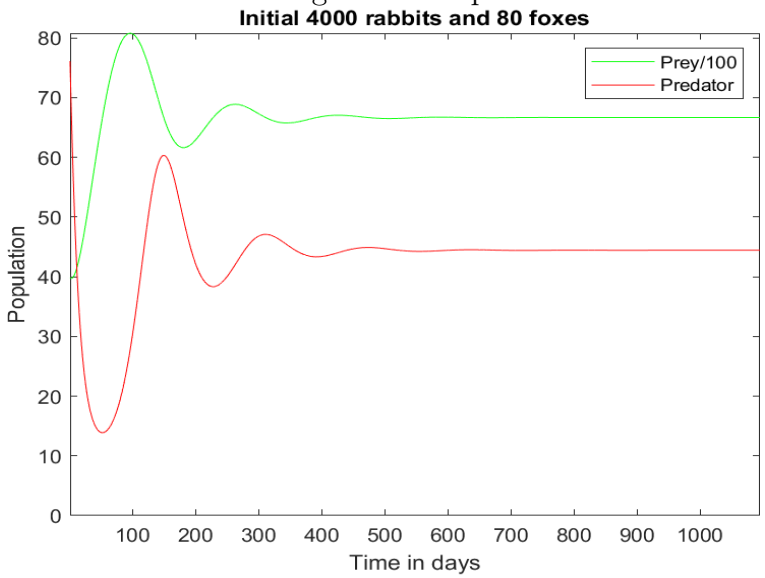
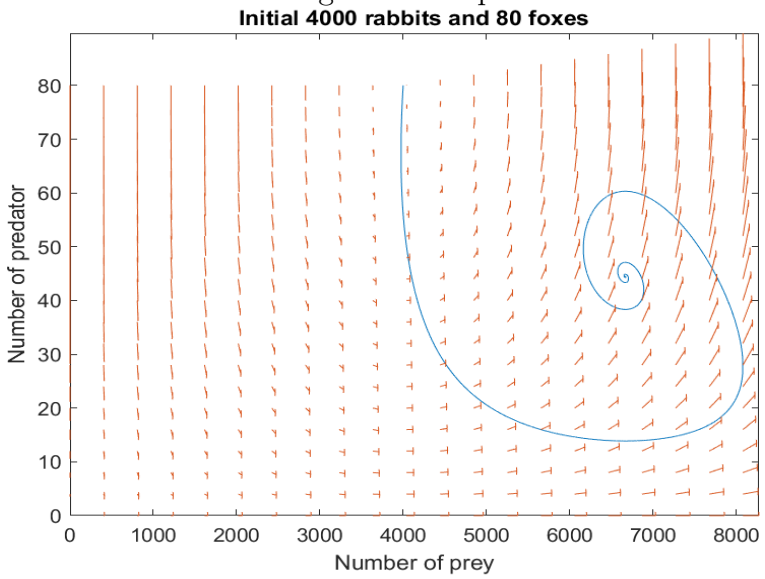


Figure 31: Population Each Other



Again, with an initial of 4000 rabbits and 100 foxes, the system behaves similarly to previous two results with different initials. The prey population ranges from around 3975 to 8079 rabbits, and the predator population ranges from 14 to 80 foxes. The system also stabilizes at 6667 rabbits(rounded up from 6666.7) and 45 foxes(rounded up from 44.4).

Since all three initial values stabilizes at the same point. I expect this point to be a stationary point. This point is stable because velocity lines are moving towards counterclockwise. I also expect two other points to be stationary points: (0 rabbit, 0 fox), (10000 rabbits, 0 fox), and both of them are stable.

3 Conclusion

We've implemented simple Lotka-Volterra model, and extended it with carrying capacity and further with predation limit. In the simple model, we observe periodic motion around the stationary point, but once we introduced carrying capacity, the system stabilized eventually. Then, we further took predation limit into account, which leads to less oscillatory motion compared to the one only with carrying capacity.

As we've discussed before, these two parameters, carrying capacity and predation limit, provide more accurate and realistic interpretation and simulation of the predator-prey interaction system.