Modeling 5

Ryan Swope

April 20, 2021

1 Roulette and Markov Chains

An oft-pondered question in the seedier casinos of the world is "Is it possible to reliably beat the house at this game?". The more mathematically and economically inclined will, of course, realize that the house always has to win in the long run for the casino to profit as a business. However, this may not have occurred to one curious gambler. He bets on 15, picks up his tonic and gin, and begins to write on his napkin as the roulette wheel spins. To simplify the problem, he decides that from here on out he will only bet on red or black, since they give him the highest chance of winning on any given game. He glances up and confirms that, indeed, this casino uses a French roulette wheel, with only one green space. He finds that his probability of winning a game is $\frac{18}{37}$, meaning his probability of losing is $\frac{19}{37}$. It's not looking good, but he decides that he should prove it rigorously so that he knows, for sure, that his best option is to leave. Based on his knowledge of linear algebra and stochastic probabilities, he is able to construct the transition matrix. His napkin now has the following matrix scrawled on it:

$$\begin{bmatrix} 1 & 1-p & 0 & 0 & 0 & \\ 0 & 0 & 1-p & 0 & 0 & \\ 0 & p & 0 & 1-p & 0 & \dots \\ 0 & 0 & p & 0 & \ddots & \\ & & \vdots & & & \\ \dots & 0 & 0 & p & 1 & \end{bmatrix} \tag{1}$$

This is an NxN matrix, where N is the number of possible states and the value represents the probability of moving to that state from the previous one; in our scenario the states are all the possible dollar amounts the gambler could have (it should also be noted that this is a very large napkin). He currently has $1000 dollars, and would like to double that amount, but he is not willing to risk more than $100 at a time. He realizes that this state can be written as a column vector, with a one in the tenth index and zeros everywhere else. He takes out his computer; this draws the ire of the pit boss and security begins to close in on the table. The gambler quickly decides that one thousand games

is sufficiently close to "long term" so he raises the transition matrix to the thousandth power. With security closing in, he multiplies this matrix by the current state, and sees his possible outcomes: 63.196634% change of losing all his money, and only 36.8029834 % chance of doubling his money. The situation is even more grim if he wants to triple his money: an 82.3502477% chance to lose and just a 17.6497523% chance to win. Having lost all hope of making money risk-free to escape his day job, he quietly walks to the casino's exit, escorted by two large men in suits. Kyle Conroy's escape from postdoctoralship will not be that easy.

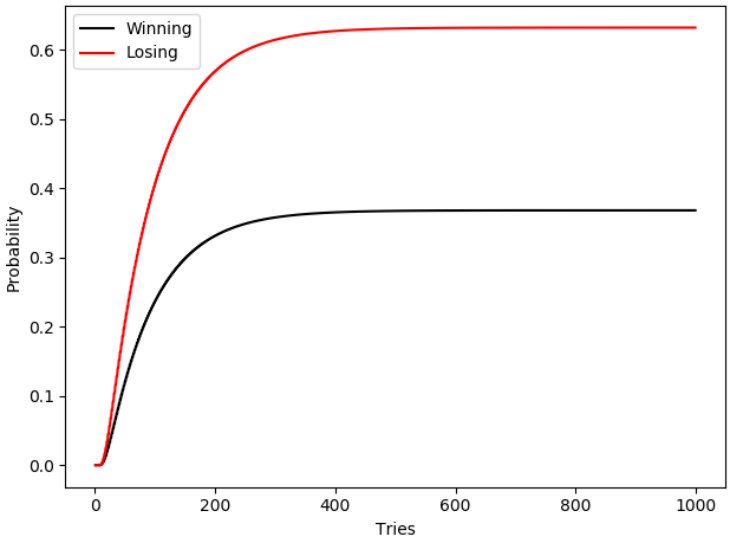

Figure 1: Chances of Kyle doubling his money after n games of roulette.

2 Discrete Epidemics

A major flaw encountered when modeling epidemics with continuous differential equations is the ability for the disease to infect and spread to fractional amount of people; in reality, the number of sick people is a discrete value. Additionally, the differential equations do no account for "random walks" — the tendency of individuals to wander around rather than stay in one position with a certain probability of getting sick. For this reason, it is appropriate to instead model disease spread by directly simulating the interactions between sick and healthy individuals.

I achieved this through the definition of two classes: `World` and `Actor`. The `World` class held the locations of all the characters, their state (healthy or sick), could place a character in the world, and could tell a character information

Figure 2: Random walk of two characters, one sick and one healthy. It runs until they meet and the healthy one becomes sick.

about the area around it (does it have neighbors, where are its neighbors, can it move to a space, etc). The `Actor` class was used to create actor objects, the characters that lived in the world. They had a state of either healthy or sick, could act (get sick if next to a sick person) or move themselves to an adjacent spot. Although the creation of these classes required significant setup, it made the implementation far simpler other implementations I've seen, and made it trivial to scale the number of actors up with the use of arrays of actors. The spread of a disease through a populations of various densities was then simulated. In all cases, there was one single patient zero, and no individuals could become healthy once sick.

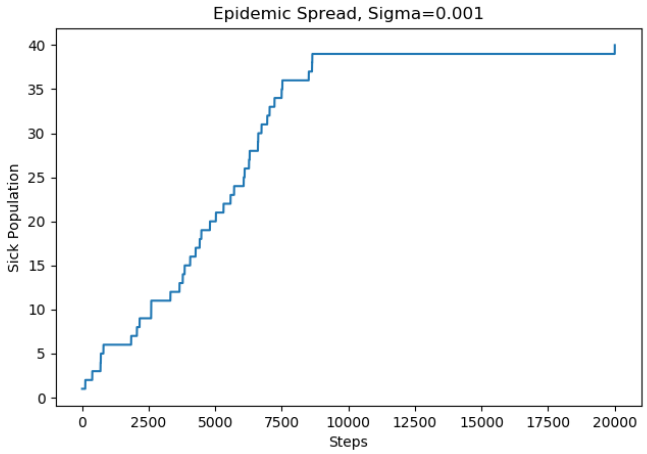

Figure 3: Epidemic Spread through population density of 0.001

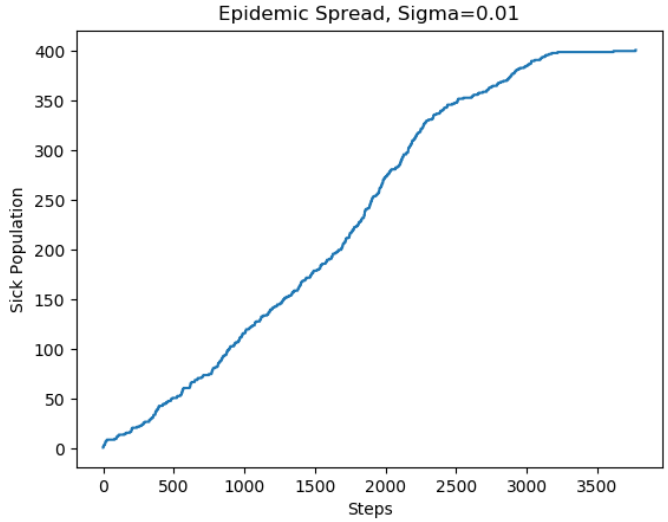

Figure 4: Epidemic Spread through population density of 0.01

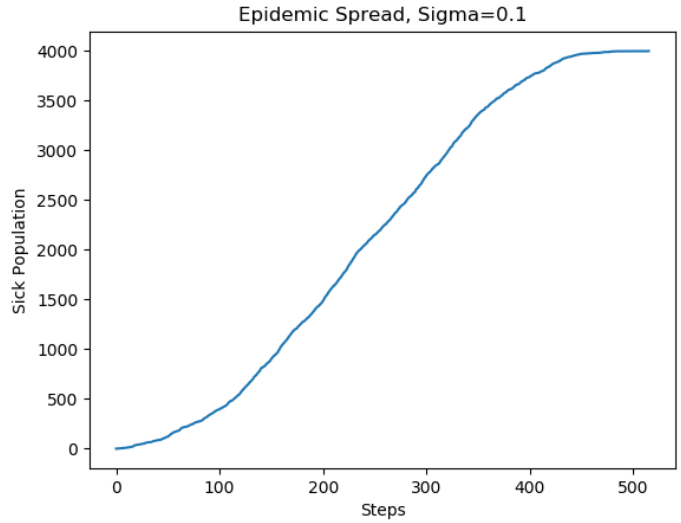

Figure 5: Epidemic Spread through population density of 0.1

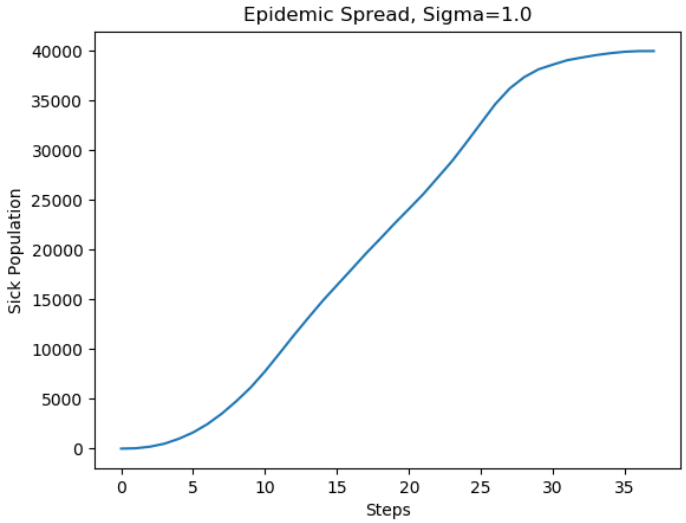

Figure 6: Epidemic Spread through population density of 1

In all these models, the rate at which the disease spreads increases at first, as more sick people means more interactions between sick and healthy people. However, there is a critical point reached where the majority of interactions are

no longer between sick and healthy individuals, but instead between sick and sick individuals. Thus, the rate at which the disease spread decreased past this point. It is clear that the less dense the population, the longer it took to infect the entire population. Plotting the number of steps against population density makes this relationship clear. The initial spike is the jump from the trivial case where the only inhabitant is the single sick person to the case where there are two people, one sick and one healthy.

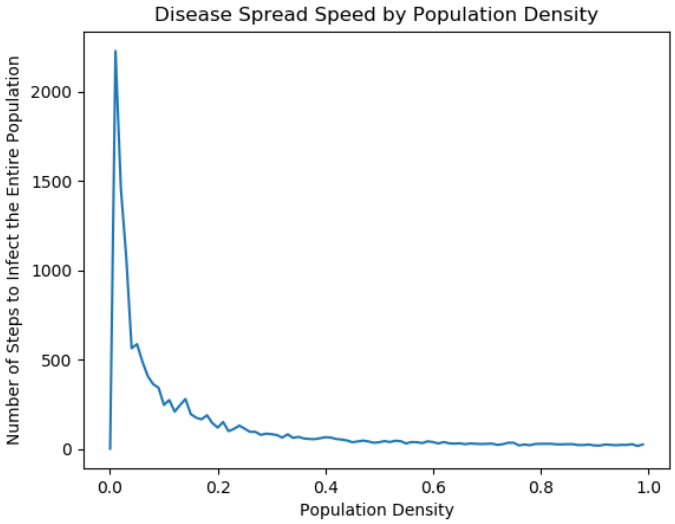

Figure 7: Disease spread speed as a function of population density

This relationship closely resembles a decaying exponential function. Indeed, when the curve is fitted with a decaying exponential, it seems that the average number of steps required to infect a given population density σ in a 100x100 grid roughly given by:

$$S = 15.23e^{-34.98\sigma+5.30} + 60.33 \tag{2}$$

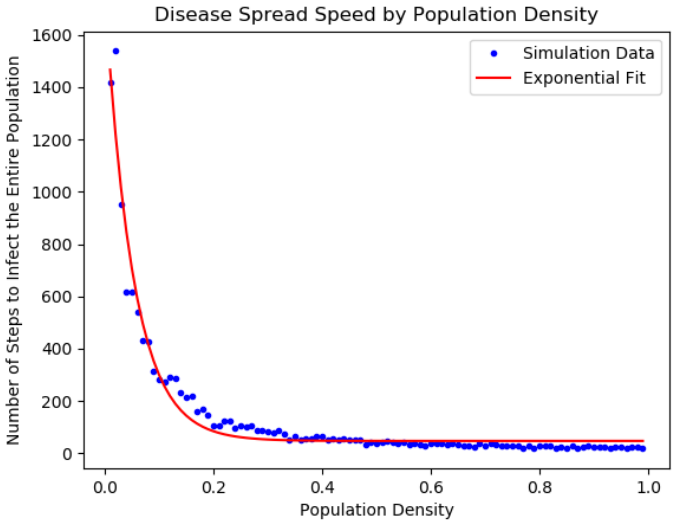

Figure 8: Decaying exponential fitted to disease spread speed

This epidemic model has its own flaws. Primarily, people move randomly and no preventative measures are taken to slow or stop the spread of the disease. It also assumes that no one is immune or recovers from the disease, so once someone has it they are infectious until the end of the simulation. We can remedy this by adding additional behaviors to our actors. For example, we can give them vaccinations that make them 50% less likely to become infected. The result is an epidemic which unsurprisingly takes longer to spread for a given σ than in an un-vaccinated population. Finally, these models assume a static population; in reality, births, deaths, immigration and emigration are all important factors in the spread of an epidemic that are not accounted for in this model.

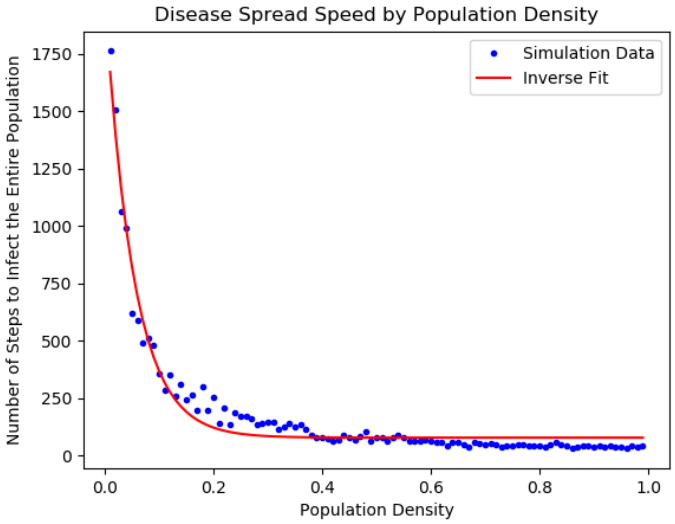

Figure 9: Speed of disease spread through vaccinated population.

3 Discrete Population Dynamics

Continuous population dynamics described by differential equations suffer from the same inequities as continuous epidemic models: species are able to recover from populations less than one. Discrete models again save us from this fallacy. I again created two classes, very similar to those created for the epidemic model: a `World` class and an `Animal` class. On top of the behaviors described before, I had to add in behaviors for the species, namely the birth, death, and predatory behaviors. To simplify the code, I threw biological accuracy out the window: a rabbit gives birth every ten steps, a fox dies if it goes ten steps without eating, and if a fox eats a rabbit it births a fox. Plotting the populations over time shows their interplay, and plotting them in phase space shows the chaotic nature of the discrete populations, and the inevitable death of at least one species.

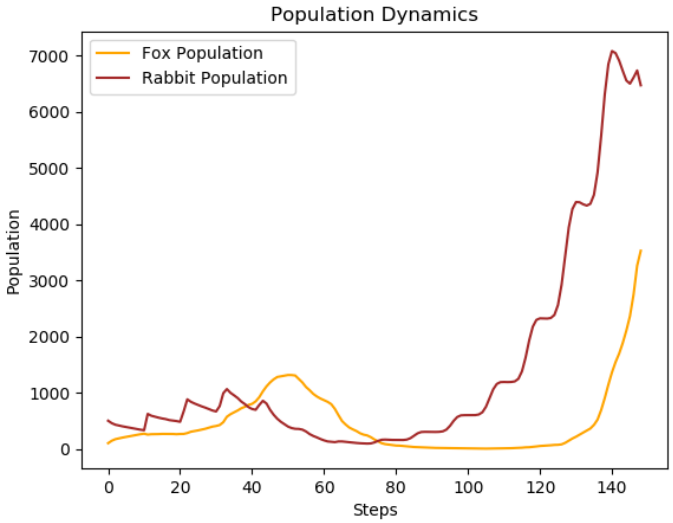

Figure 10:

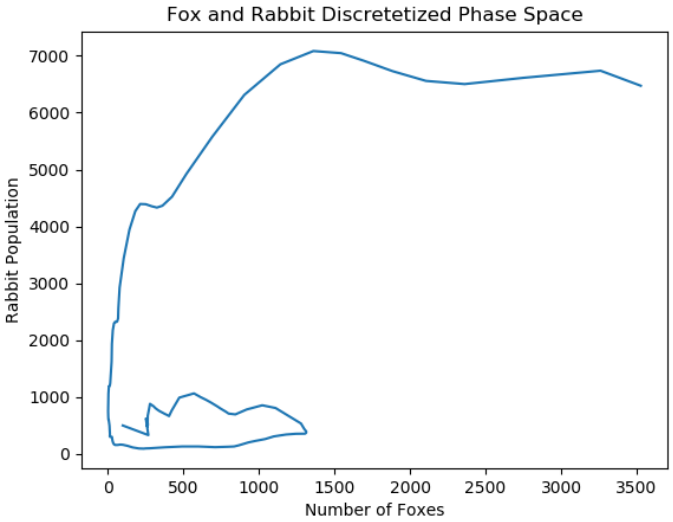

Figure 11:

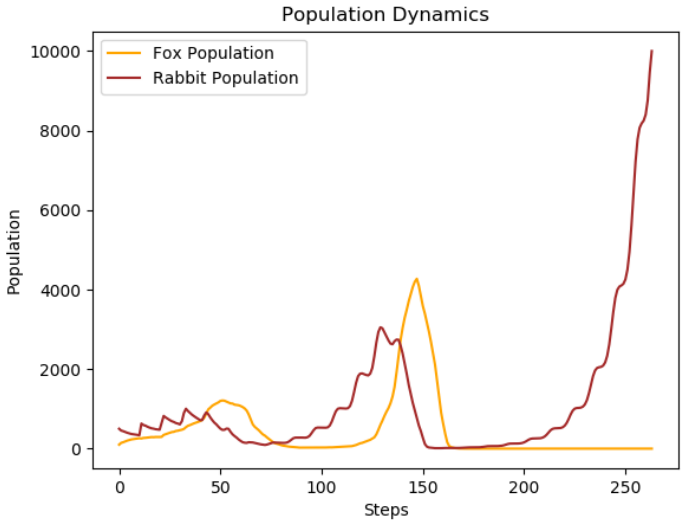

Figure 12:

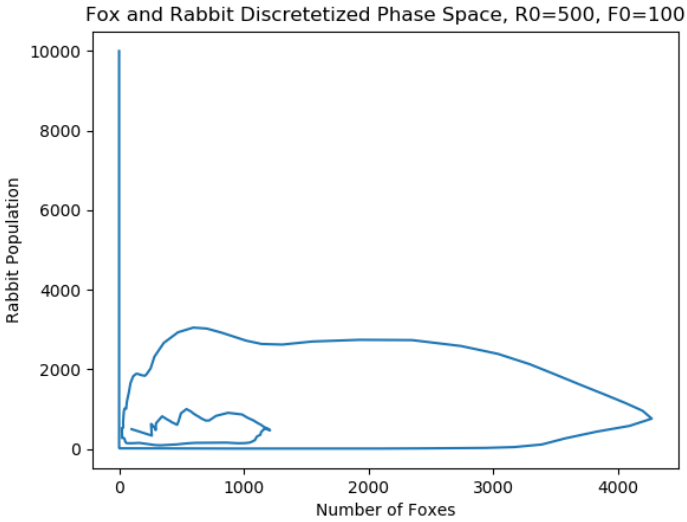

Figure 13:

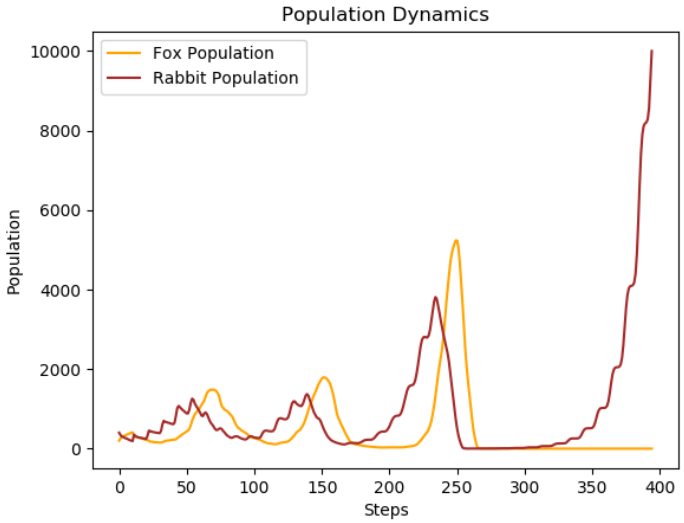

Figure 14:

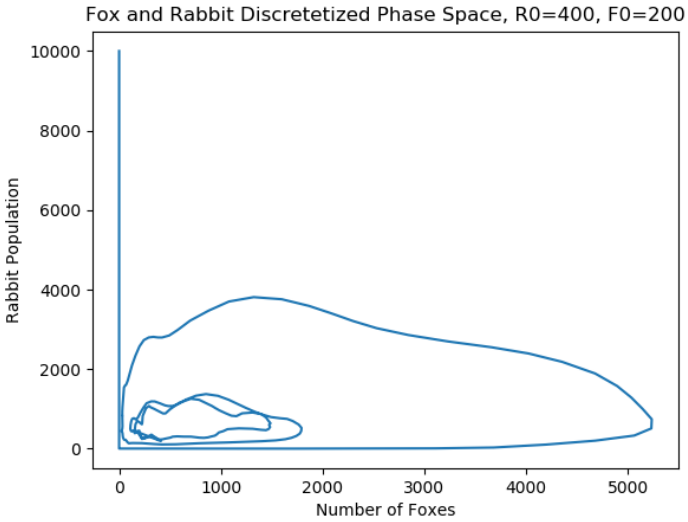

Figure 15:

Even in cases where the species seem to be very near a stationary point, the populations quickly move away. Though the populations behave in a vaguely orbital way on small time scales, on the whole they clearly do not map out any

kind of stable orbit. There is no characteristic time associated with these paths that I could discern.

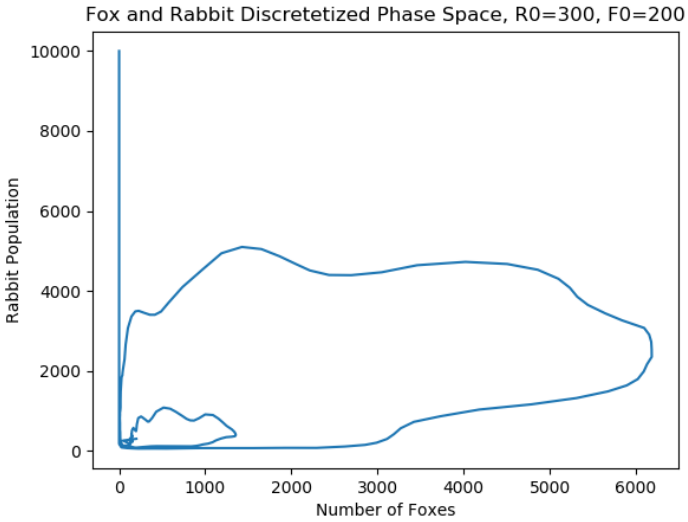

Figure 16: Phase plot where the species almost reach a stationary point.

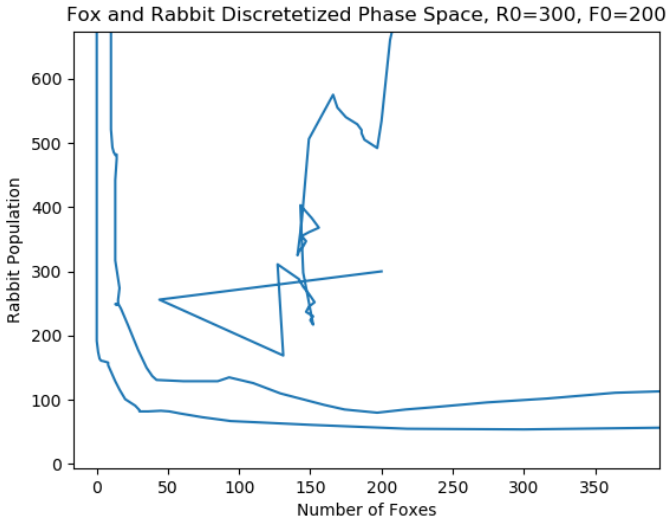

Figure 17: Closer view of near stationary point.

I then added a third species, wolves, which eat both foxes and rabbits. Similar to foxes, wolves starve if they do not eat, and when they eat a fox or a rabbit they birth a wolf. Interestingly, the wolves often helped control the fox population, so the three populations were able to coexist longer than just the two together. These populations were plotted against time, and in phase space using two separate curves.

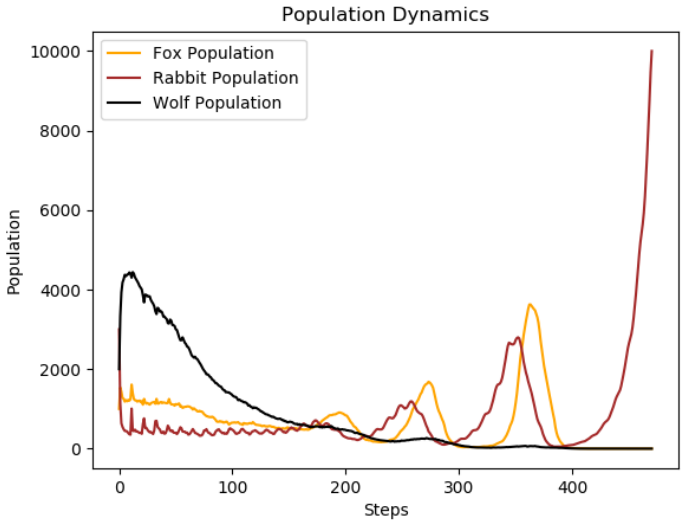

Figure 18:

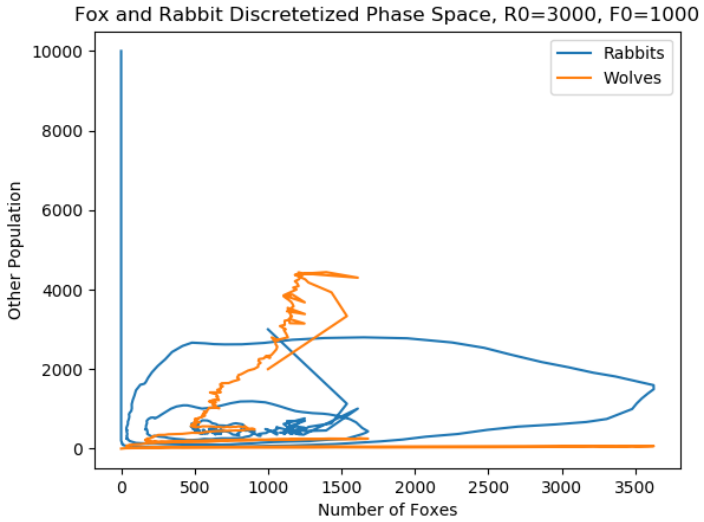

Figure 19:

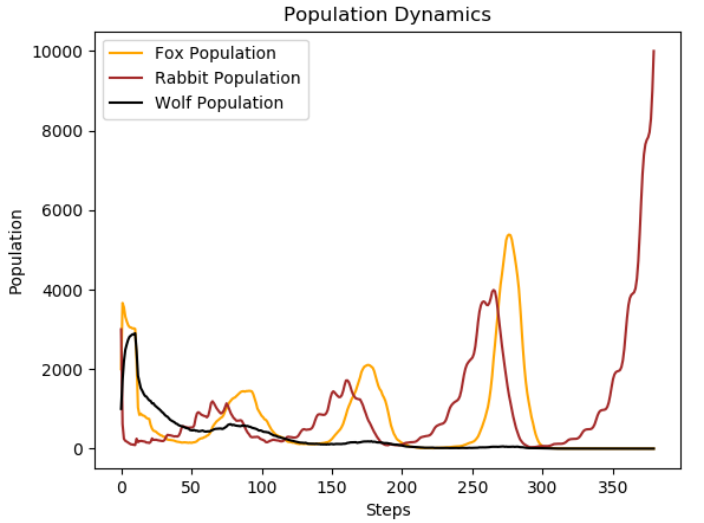

Figure 20:

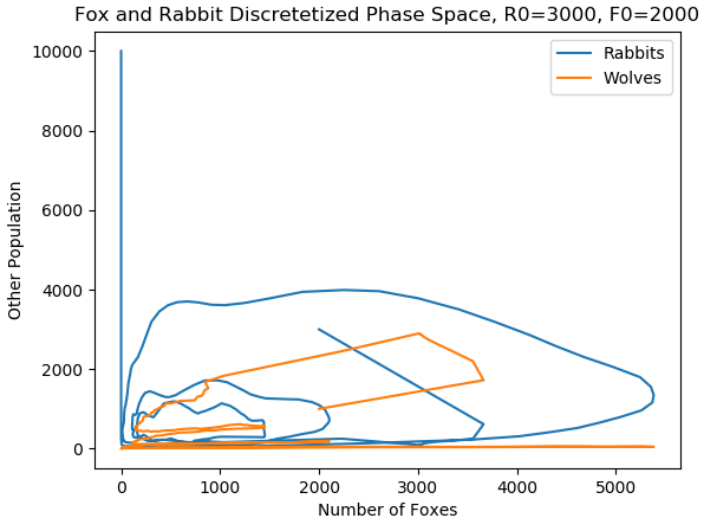

Figure 21:

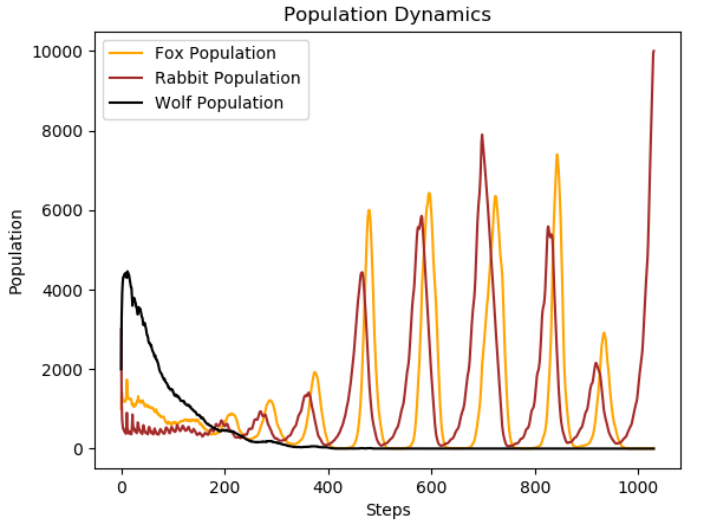

Figure 22:

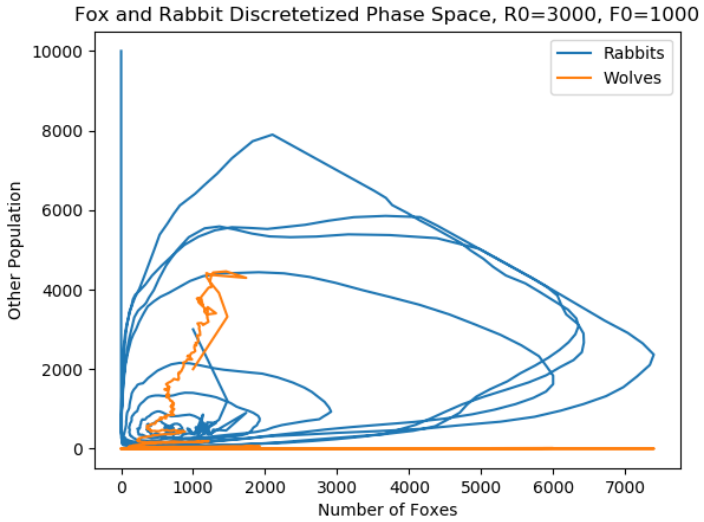

Figure 23:

As you can see in 23 the species reach a point that is very nearly a stable orbit, but eventually falls apart with the starvation of both predatory species. In addition to these phase plots, we can plot the Rabbit, Fox, and Wolf populations in three dimensional phase space.

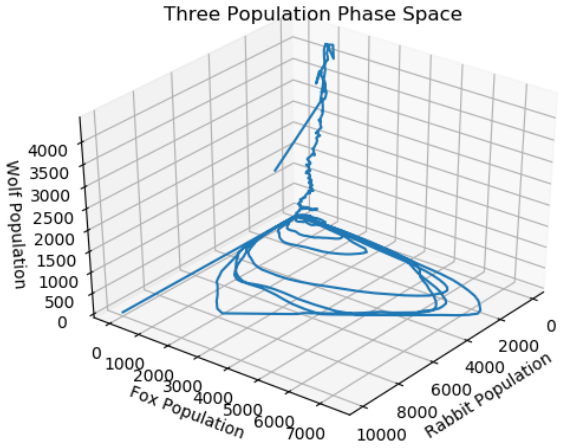

Figure 24: Three dimensional phase space.

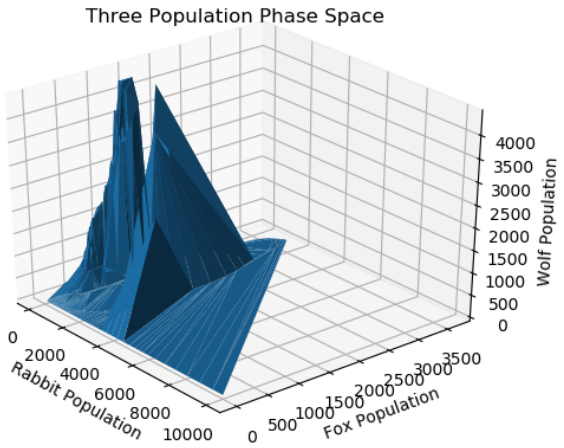

Figure 25: Three dimensional phase space with surface.

I decided to try to simulate a population of humans and zombies. The rules were simple: the humans would give birth every so often, and have a chance of killing a zombie if they encountered one. The zombies would have a chance to eat a human if they encountered one, turning them into a zombie. I thought I may be able to find a point at which humans are able to survive the zombie outbreak. However, I was unable to find such a point. Humanity, it seems, is as doomed as ever.

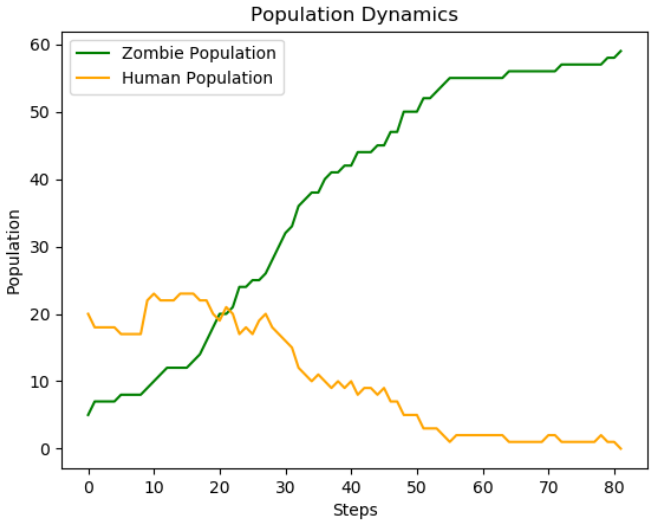

Figure 26: Zombie outbreak population dynamics.

These population models are not without flaws either. The choice of birth and starvation rates was fairly arbitrary and primarily chosen to be long enough for a proper simulation but not too long that the simulation would take an unreasonable amount of time to run. There was also assumed to be an infinite supply of "carrots" for the rabbits to eat, so that they never starve or die a natural death. In reality, world would have a carrying capacity for the rabbits, and they would begin to die if there were too many of them. Additionally, the choice of predatory and reproductive behavior was a choice made with computational simplicity and efficiency in mind. If a species gives birth when two members are next to each other, the simulation takes far longer to run, and the behavior is extremely difficult to control.

4 References

All the code used in this project was built by me completely from scratch. I utilized documentation on docs.python.org to assist in learning about classes and their implementation.