Project 1

Predator/Prey Dynamics

Matthew Lopez Tarsky
[Redacted Member #1]
[Redacted Member #2]

ENGR 285

Objective #1: The types of outcomes the simulation can have, and how common they appear to be. (Not just for the default parameters)

From our testing, we concluded that simulation can have three outcomes

1. **Both fish and sharks die**

This can occur due to the sharks eating all the fish, and when there is no food, the sharks cannot reproduce faster than they can die, thus dying out too.

2. **Fish live, Sharks die**

This can occur because the sharks die out before eating all the fish, allowing the fish to grow nearly exponentially.

3. **The population alternates or resembles the Lotka-Volterra Model.**

This outcome resembles a result similar to the LVM, in which the "high points" and the "low points" of each population resemble the model, just not in sync with each other. Though we may have enough time or steps, this can lead to either of the two other outcomes mentioned before, but this is uncommon and occurs usually due to randomness.

4. **Sharks live, Fish die**

But how common do they seem to appear in the simulation?

When running the simulation multiple times with different parameters, time is a very huge factor. If given enough time, almost always, sharks will go extinct; that is because the sharks' population is dependent on the fish, and that makes it the most common outcome.

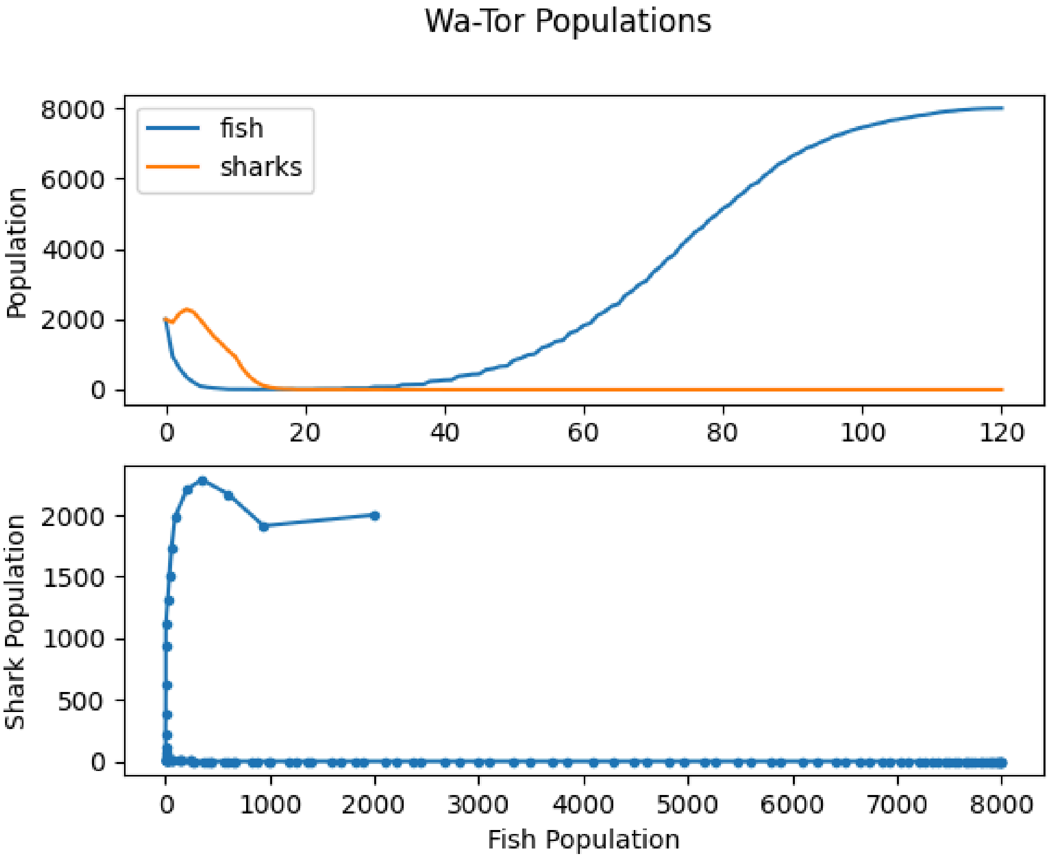


Figure 1: Breed time: ***3****, Energy gain:* ***4****, Breed Energy:* ***10****, Dimensions:* ***100x80****, Initial Fish:* ***2000****, Initial Sharks:* ***2000****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

Along with that, the outcome that is almost equally common is both species surviving. In these instances, the outcome model will be the LVM. This outcome will not show exactly which species dies or survives first, but it will show how the population fluctuates over time.

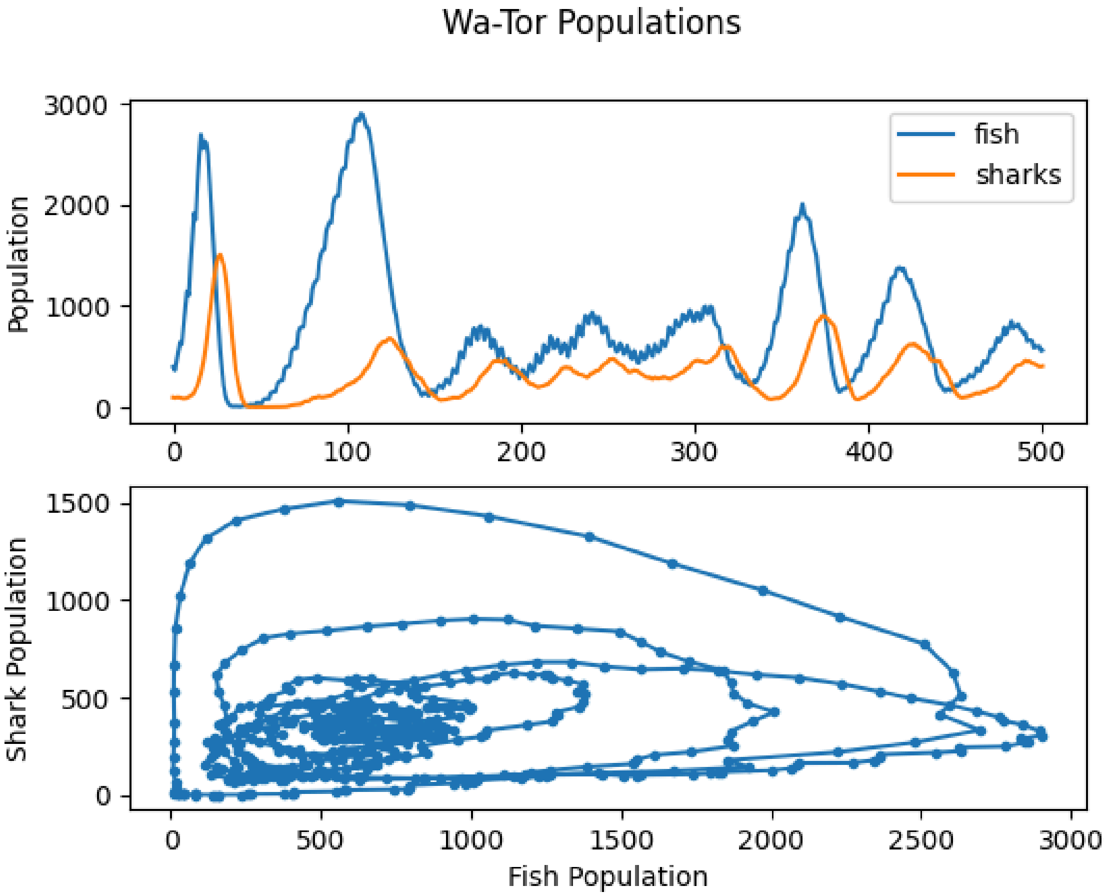


Figure 2: Breed time: ***3****, Energy gain:* ***4****, Breed Energy:* ***10****, Dimensions:* ***100x120****, Initial Fish:* ***400****, Initial Sharks:* ***100****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

The second rarest outcome is where both species go extinct, that is because it is a very time-dependent outcome. This outcome will usually only happen if the time parameter is changed to a large enough value that the fish will go extinct after being completely hunted down, and then the sharks will eventually all die out as well.

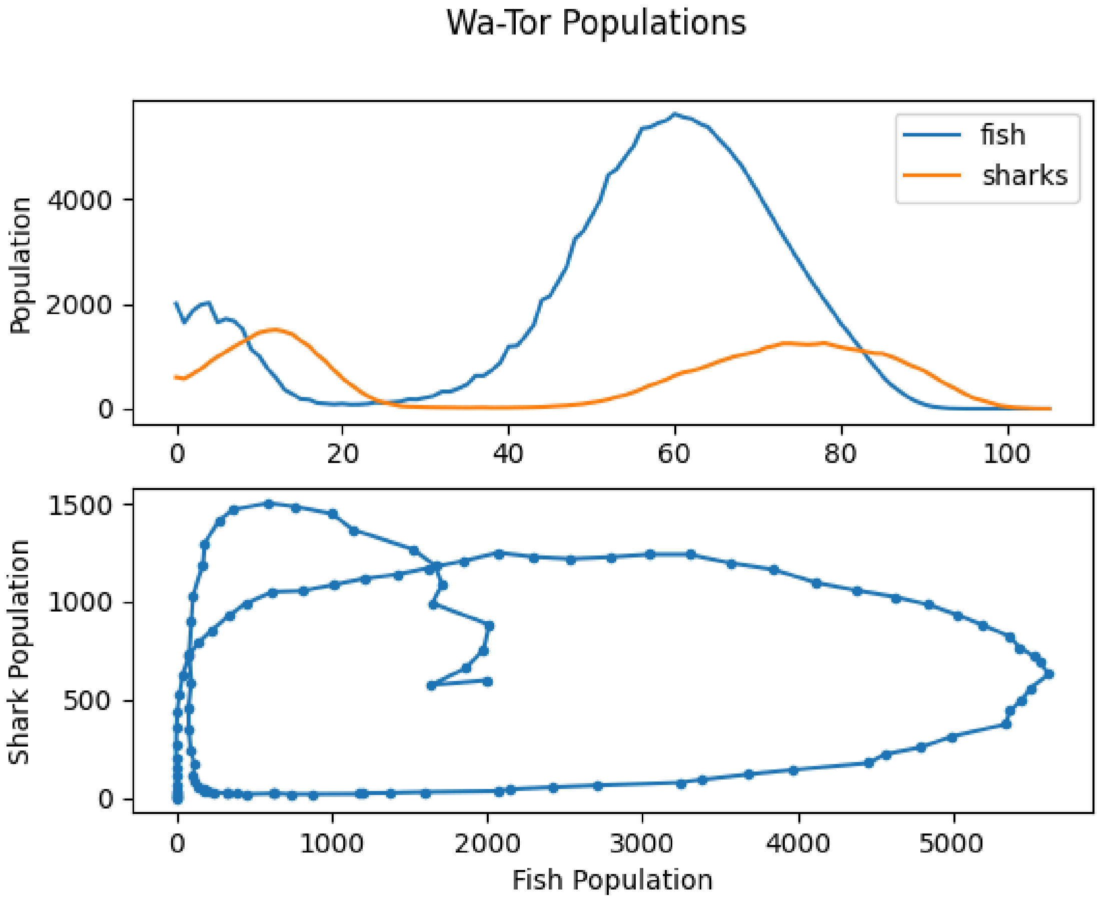


*Figure 3: Breed time: **3**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **600**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

The rarest outcome of all is that fish will go near extinct and stay there. That is because this outcome is reliant on both the shark's population and time; basically, the sharks reproduce faster than they can die. This outcome will appear almost only when given enough time exactly where sharks will eat all the fish, and the shark population grows. So, as we see, the more specific the parameters have to be for an outcome, the more rare that outcome is.

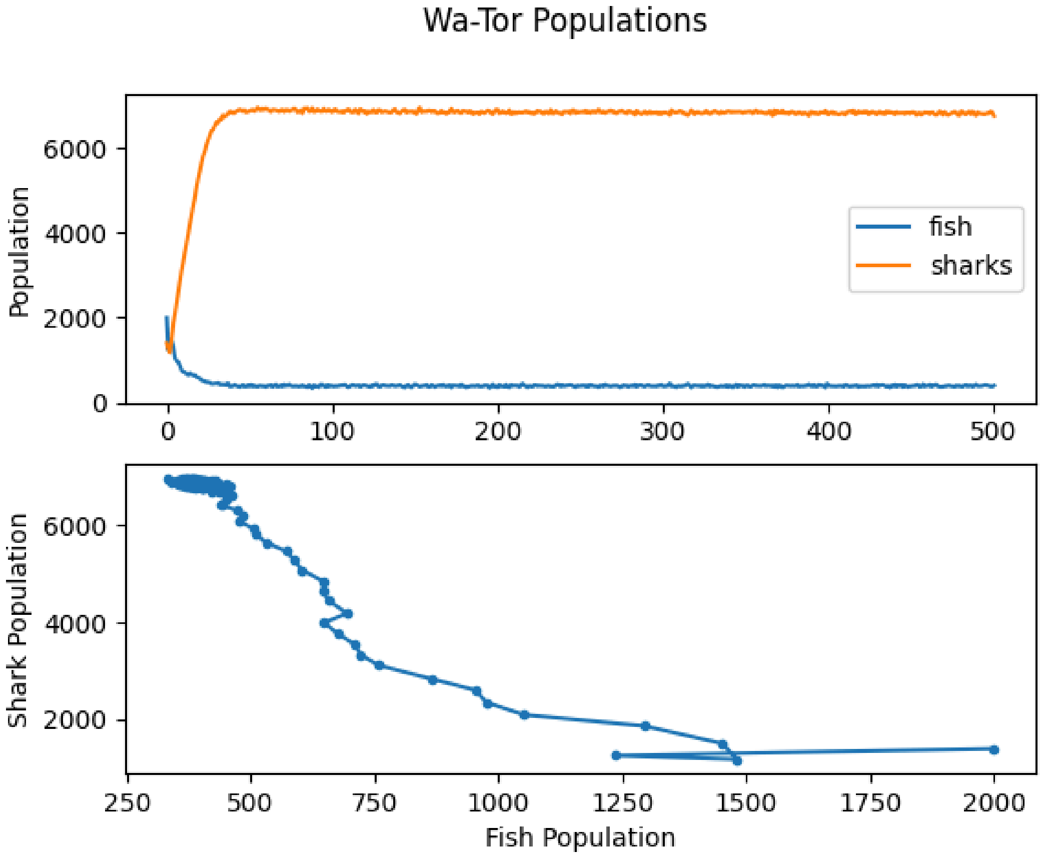


*Figure 4: Breed time: **3**, Energy gain: **4** Breed Energy: **5**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **1400**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

Objective #2: The conditions (i.e. parameters) under which it appears the Lotka-Volterra Model models the simulation results well; i.e. when the populations appear to follow trajectories that resemble those of the Lotka-Volterra Model.

The Lotka- Volterra model graphs both of the populations to fluctuate as sinusoidal waves, with the predator and prey populations being off sync from each other by approximately 90 degrees or $\pi/2$ if they were to model with a sine wave function; if the populations were modeled using a cosine and sine function. In other words, when the population of one is at a “high” point, the population of another is at a “low” point.
It should look similar to the image below.

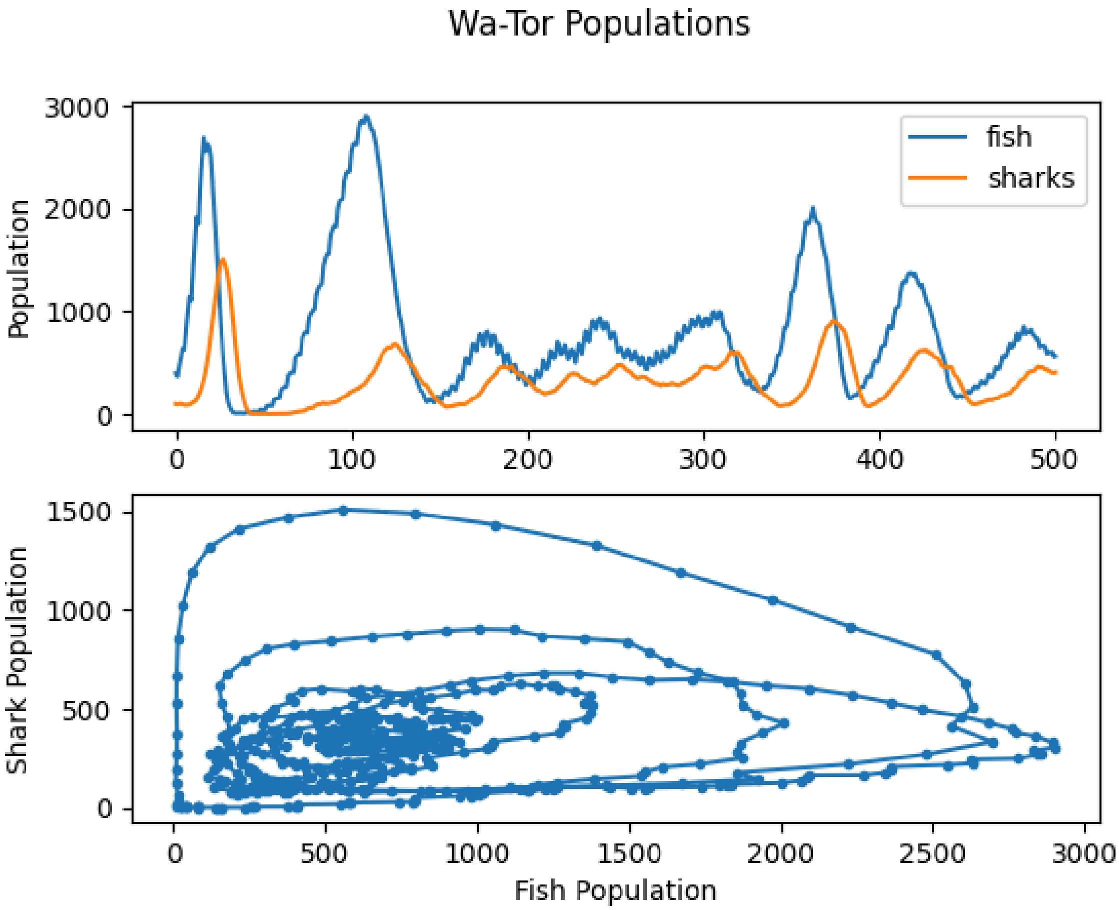


Figure 5: Breed time: ***3****, Energy gain:* ***4****, Breed Energy:* ***10****, Dimensions:* ***100x120****, Initial Fish:* ***400****, Initial Sharks:* ***100****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

The high/low points of the population seem to be off-synced. The populations of fish and shark populations have maximums that all seem to go down over time, with their equilibrium being their oscillation point. Realistically the maximums/minimums of both populations should stay consistent, but don't change over time like above.

To figure out which set of parameters was going to be ideal, we would isolate each variable by keeping all the other parameters constant, determine which conditions matched the behavior of the model, and then try to mix them in. We can also use our knowledge of how the Lotka-Volterra Model works. We know that the model assumes:

- The population, along with time, is continuous.
- When the population of predators is low/nonexistent, the prey has a dramatic growth spur.
- When the population of predators is low/nonexistent, the predator has a dramatic decay spur and can even reach zero.
- The rate at which the prey population decays, is proportional to the population size of the predator.
- The rate at which the prey population grows is proportional to the population size of the prey.

We can use this to predict how a parameter would affect it.

There are some parameters that we can excuse not to consider to have an effect. These parameters include the start energy, which only affects the beginning. We can also exclude the setup as making it false or a nonbasic setup because to properly help model the model, the simulation would have to be random. Finally, we can ignore the number of steps as long as we kept substantial enough because if the conditions modeled the model well, the simulation would never terminate and would go on for an infinite amount of time. This would be a condition we might have throughout the trials. When deciding on some parameters match model, we decided based on how often the simulation deviates to the two other outcomes, a lower deviation, means a better match/consistency.

To start with, we decided to start with an equal population of both species:

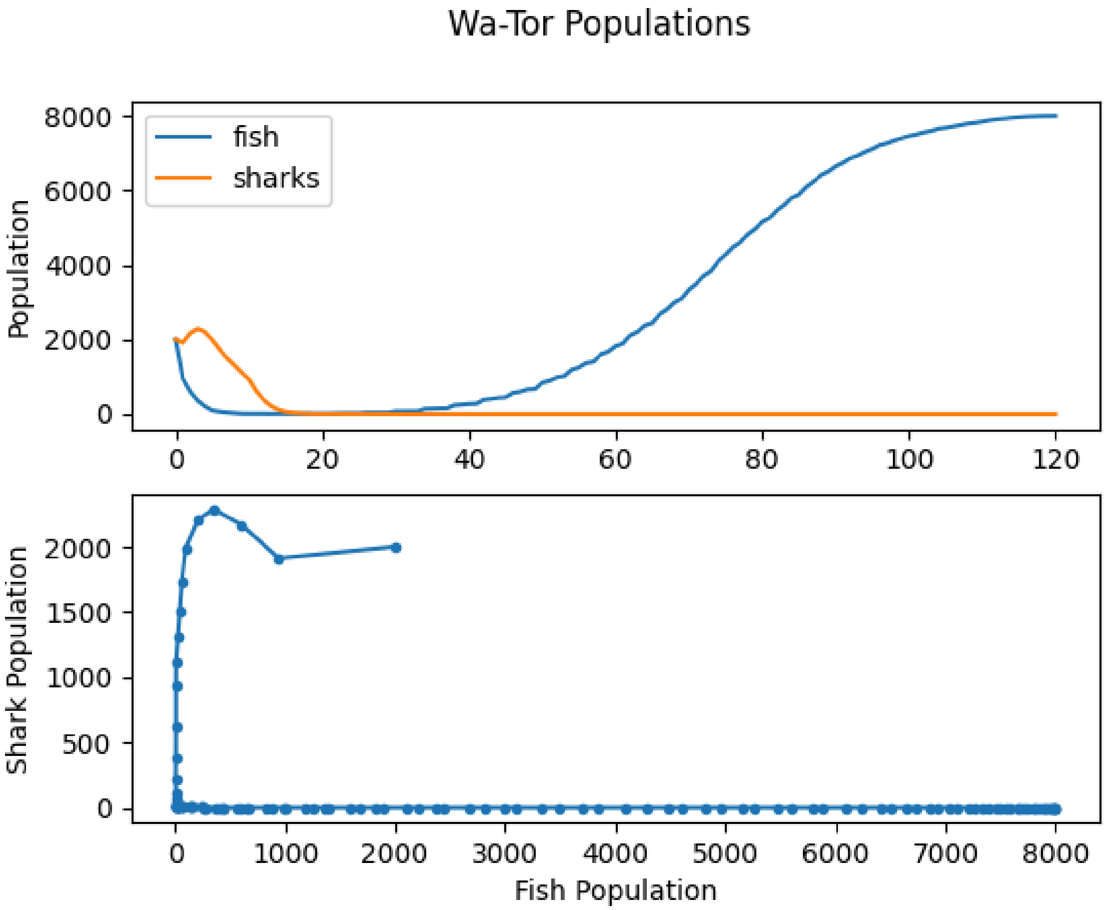


*Figure 6: Breed time: **3**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **2000**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

We found that the sharks most of the time would die out and lead to unlimited growth of the fish population, which matches one of the assumptions of the model.

When using the program, we found that when the population of the fish was substantially higher than the population of the sharks, it brought a better correlation with the model as seen below.

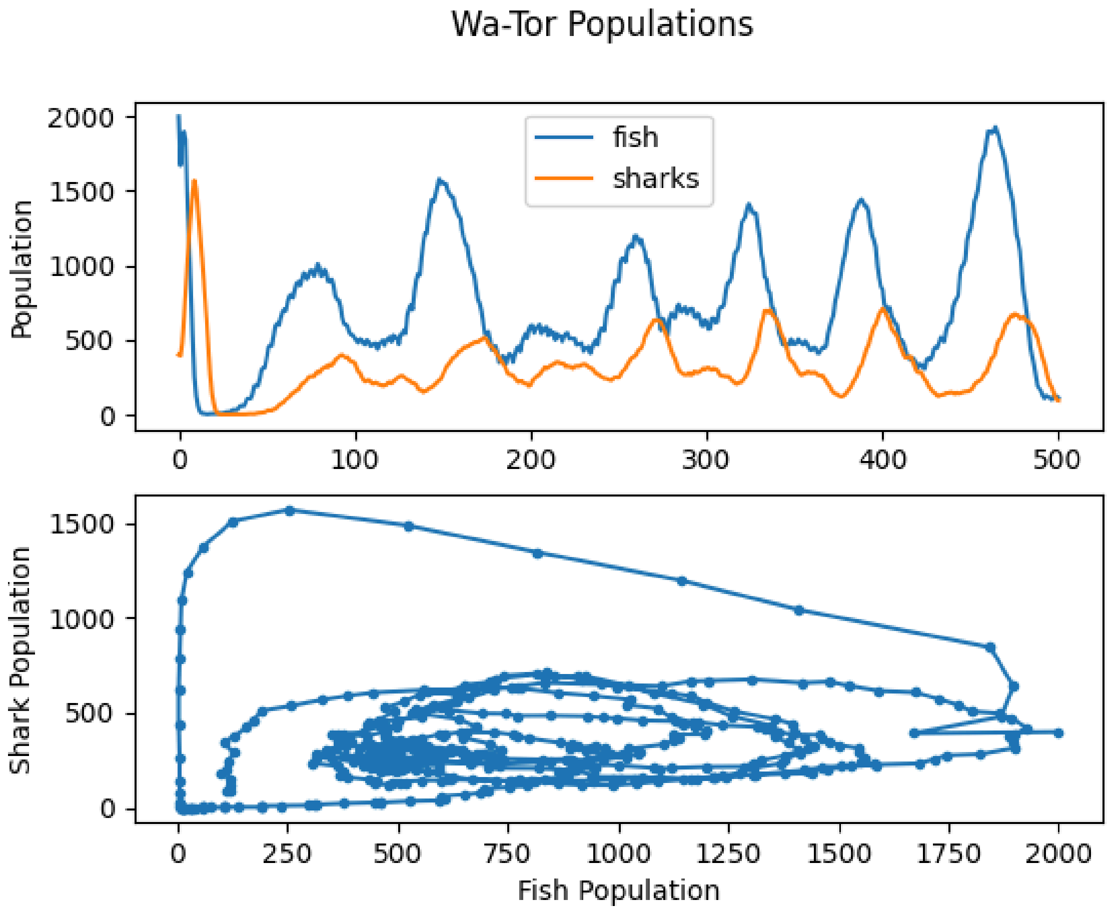


Figure 7: Breed time: ***3****, Energy gain:* ***4****, Breed Energy:* ***10****, Dimensions:* ***100x80****, Initial Fish:* ***2000****, Initial Sharks:* ***400****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

Of course, to increase the population of the fish, there needs to be enough room, so the dimensions of the simulation also have to increase along with it. When making the length of each dimension “close in value,” we found that it slightly increased in the “correlation” of the model.

For example, looking at the two images below, if we were to elongate one side of the area, we found that most of the time the outcome would result in the fish having immense growth. This is most likely because since the fish are more spaced out, the sharks need to move more.

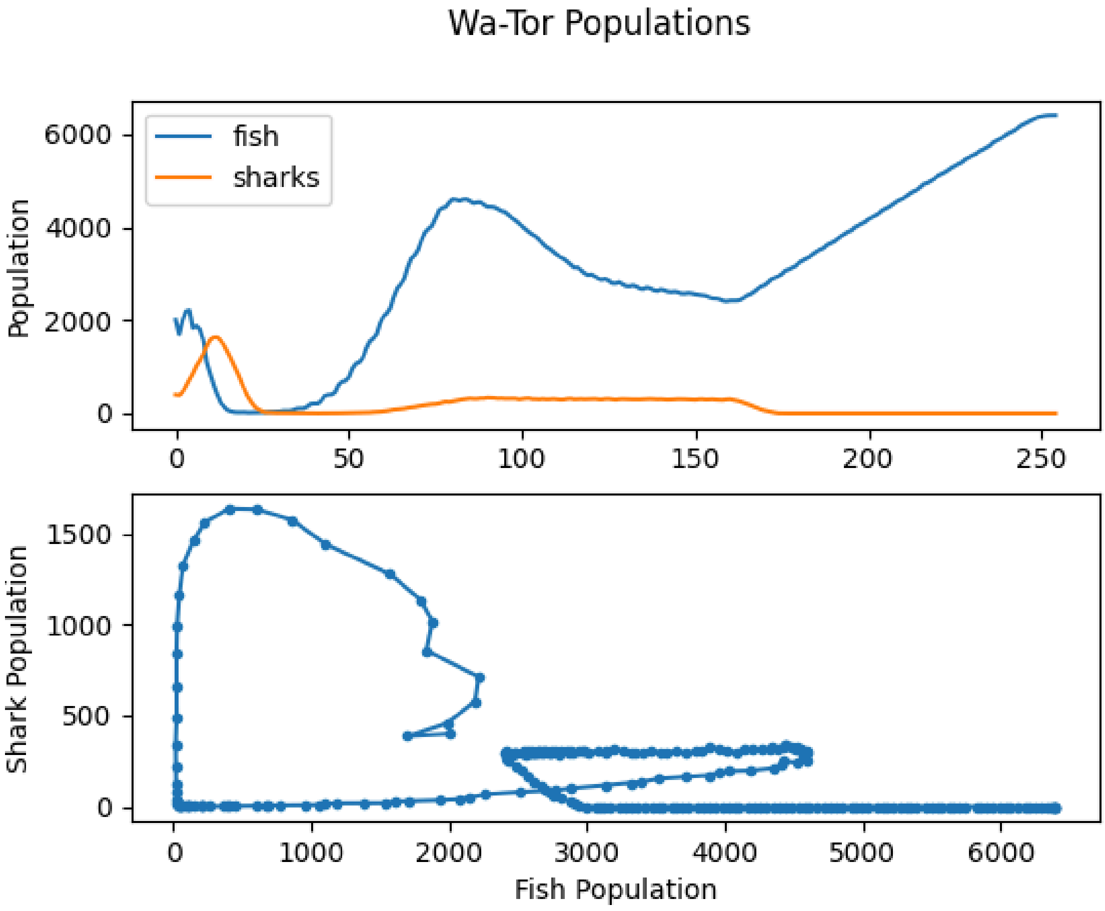


*Figure 8: Breed time: **3**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **40x160**, Initial Fish: **2000**, Initial Sharks: **400**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

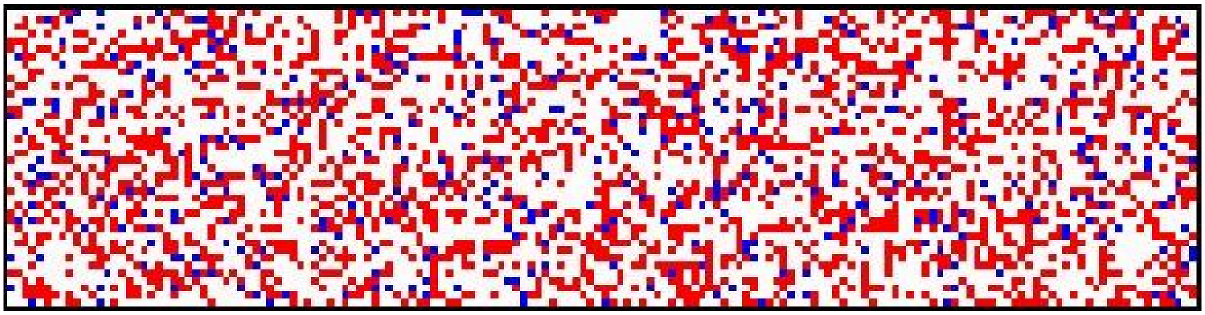


*Figure 9: Breed time: **3**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **40x160**, Initial Fish: **2000**, Initial Sharks: **400**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

Continuing on with the testing:

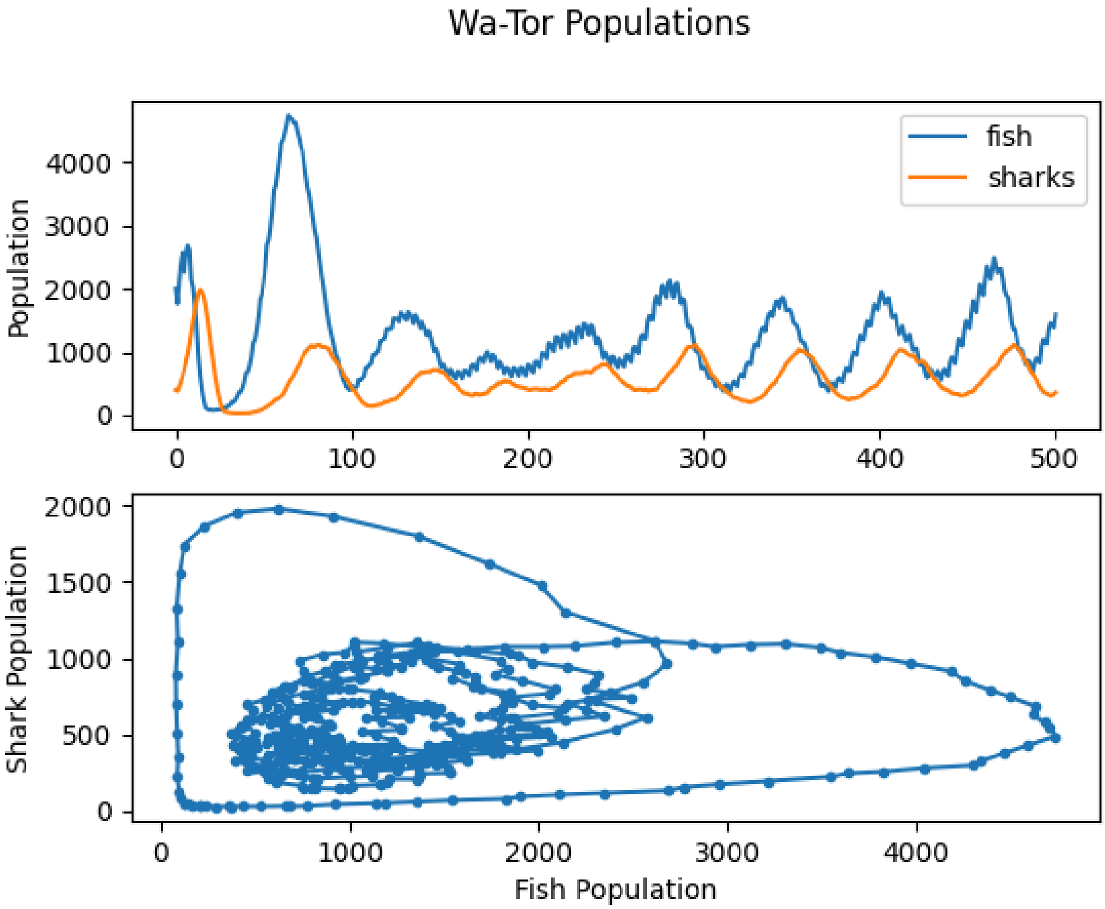


Figure 10: Breed time: ***3****, Energy gain:* ***4****, Breed Energy:* ***10****, Dimensions:* ***100x80****, Initial Fish:* ***2000****, Initial Sharks:* ***400****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

Looking at the graph above, the beginning of the simulation shows that fish and shark populations quickly fluctuate, but eventually they reach an equilibrium state. This equilibrium state can be described as the sort of “carry capacity” of the simulation, with the fish staying at their initial population, while the sharks seem to be above their original starting position. Though with the setup above, the results were inconsistent, In order to attempt to address this inconsistency, we increased the shark population by a bit, which seems to change the outcome of the population as seen below.

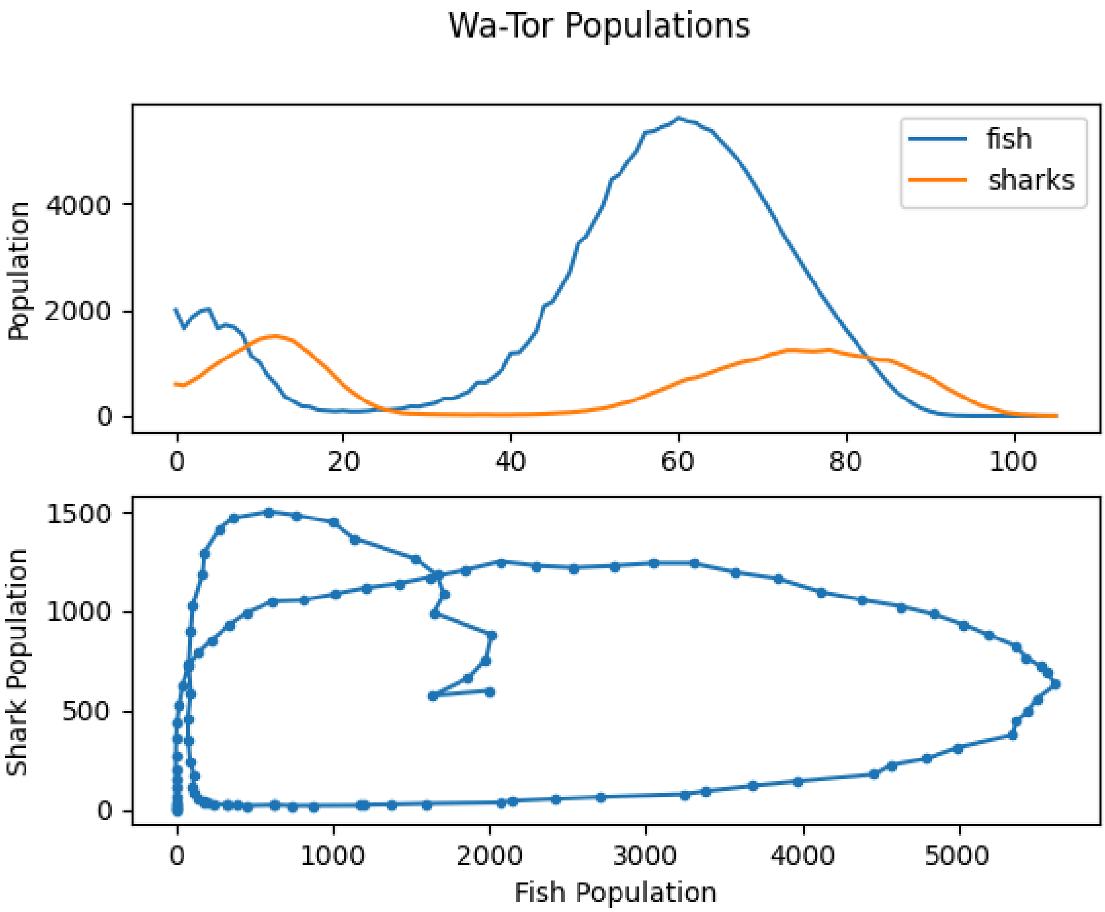


*Figure 11: Breed time: **3**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **600**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

When the initial shark population comes closer to the initial fish population, the outcome of simulation extermination of both, in which the fish population reaches zero, which means over time, the shark population plummets to zero.

We assumed this was due to the previous reason of the graph before, which was due to inconsistencies or randomness, meaning it doesn’t match the model. So we continued messing around with the simulation.

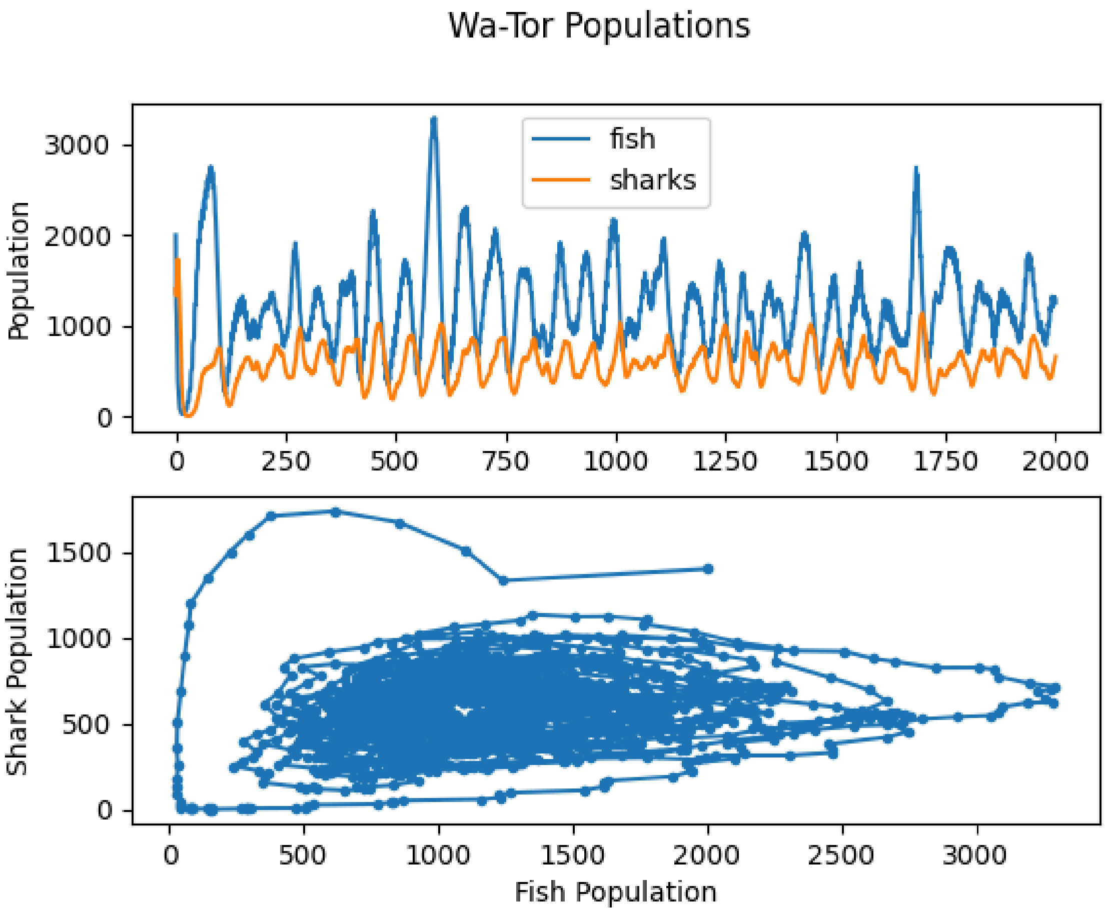


*Figure 12: Breed time: **3**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **1400**, Steps: **2000**, Start Energy: **9**, Setup: **Basic***

We found that a population of around the range of 1400 sharks and 2000 fish at the dimensions listed in the image above, brought the most consistent results. Though this result can vary and we suspect it seems like some kind of fish-shark ratio like around 1 fish for every 3/4 of a shark for this case, would be ideal. There is also the dimensionality which would be required to be increased along with the population because the more individuals there, the more space that is required. As we saw before, if the total population is too dense or too light, then the results begin to differ from the model. So there is probably a total population per unit area (population density) along with the fish-shark ratio we mentioned earlier.

Next, let's test this scenario using the main parameters. For this, we'll be keeping every variable constant except for at least one of the main parameters. Before beginning, we predict that increasing each main parameter will cause either both populations to be exterminated or the fish to dominate to be more likely.

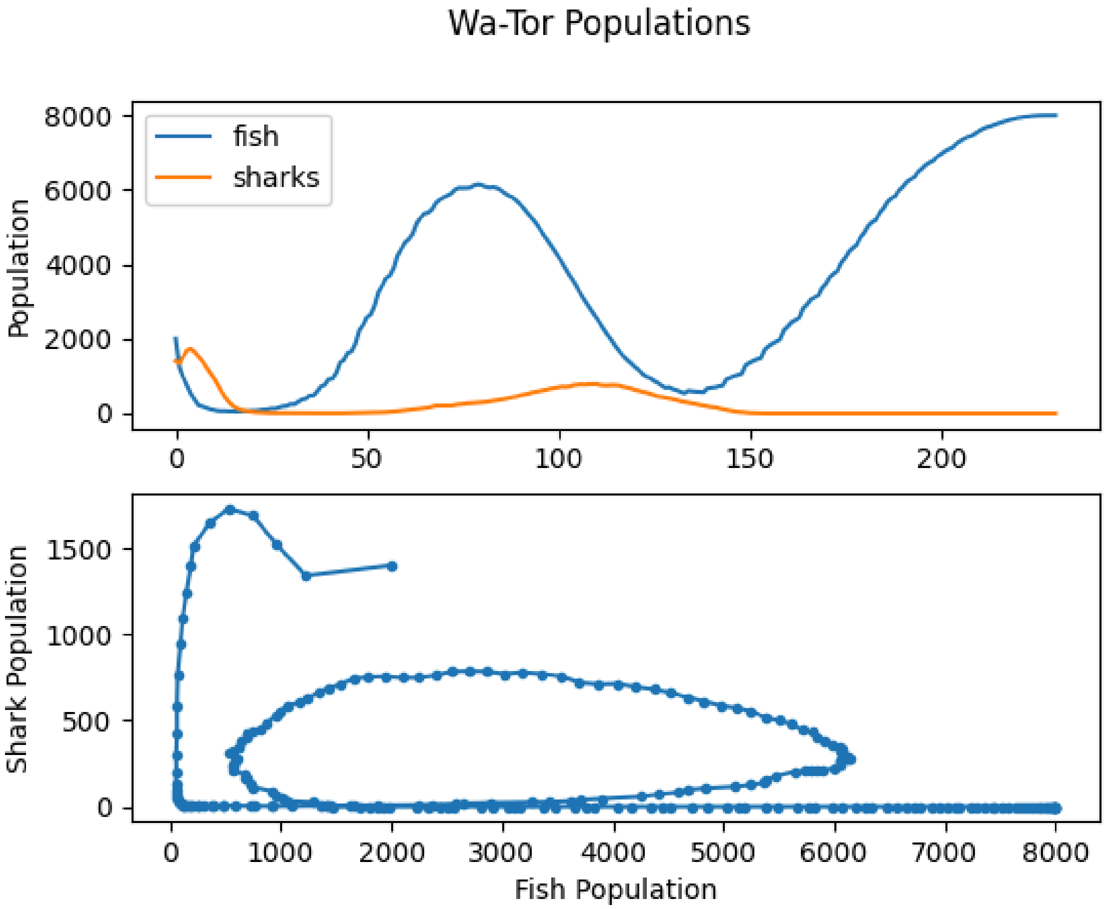


Figure 13: Breed time: ***4****, Energy gain:* ***4****, Breed Energy:* ***10****, Dimensions:* ***100x80****, Initial Fish:* ***2000****, Initial Sharks:* ***1400****, Steps:* ***1000****, Start Energy:* ***9****, Setup:* ***Basic***

Increasing the breed time of the fish seems to cause the simulation to overall cause it to immediately deviate from the model. It would seem that the breeding time of the fish being increased causes the shark population to not have enough thus dying out before the fish and thus the fish thrive. Increasing the breed time by just one already causes the change in the simulation from the model

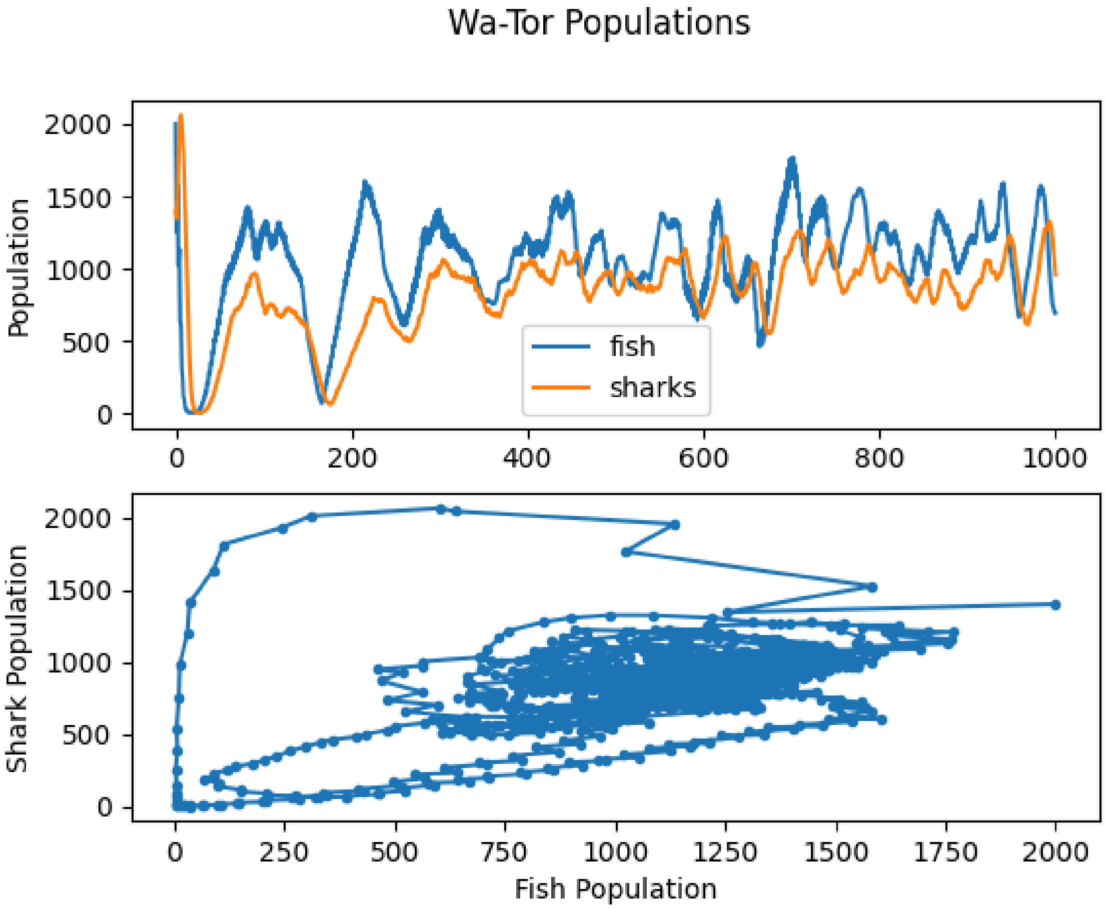


*Figure 14: Breed time: **1**, Energy gain: **4**, Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **1400**, Steps: **1000**, Start Energy: **9**, Setup: **Basic***

Decreasing the breed time seems to have a better correlation with the model, compared to when we increased it. So we concluded that having a lower breed time matches the model more clearly.

Next, let's test the energy gain of the sharks after eating the fish by increasing the parameter one by one. We suspect that this will lead to the outcome in which they are exterminated. As giving more energy to the sharks grants them moves, it will allow them more time to pursue the fish and thus the fish a lower survival rate.

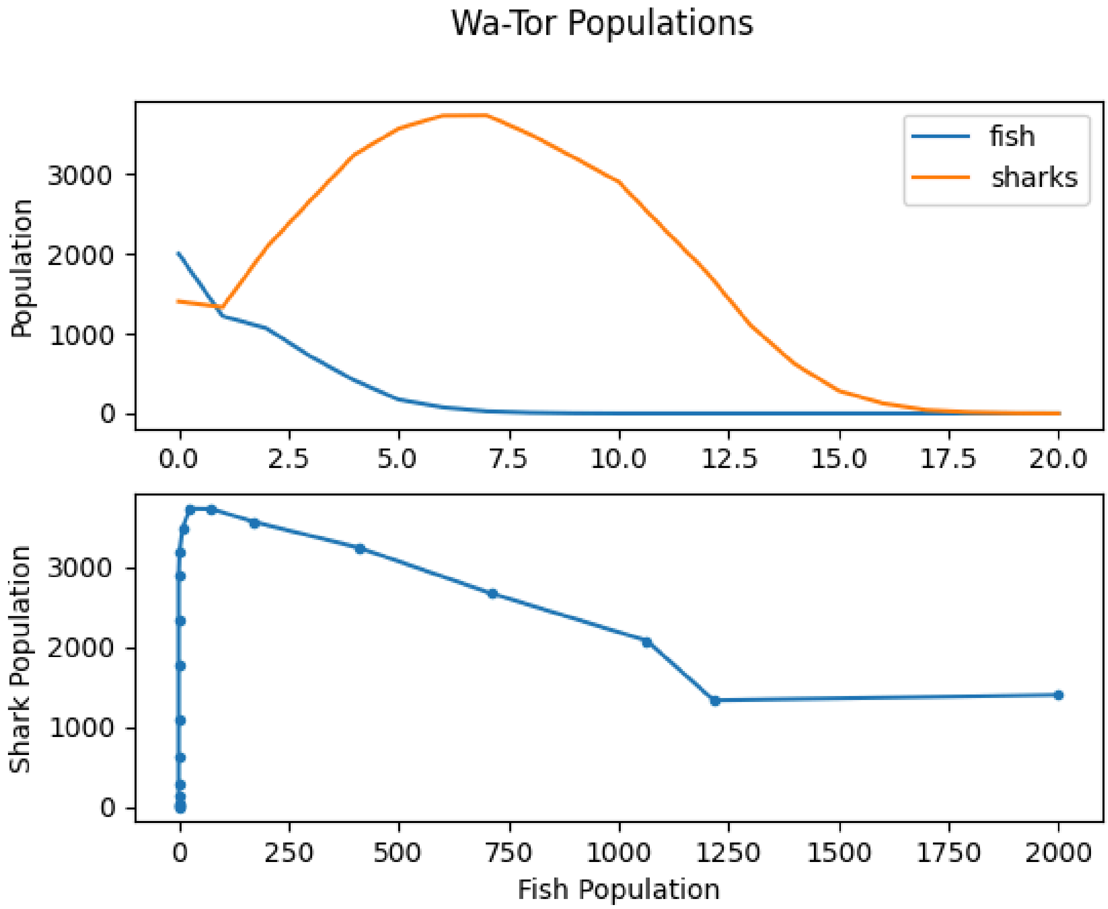


*Figure 15: Breed time: **3**, Energy gain: **11**, Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **1400**, Steps: **1000**, Start Energy: **9**, Setup: **Basic***

As we suspected, the population of sharks grows fast, which leads to the fish quickly. Increasing the energy gain too much increases the growth of sharks, which over-populates the fish population, causing them to both die out due to increased predation and lack of resources.

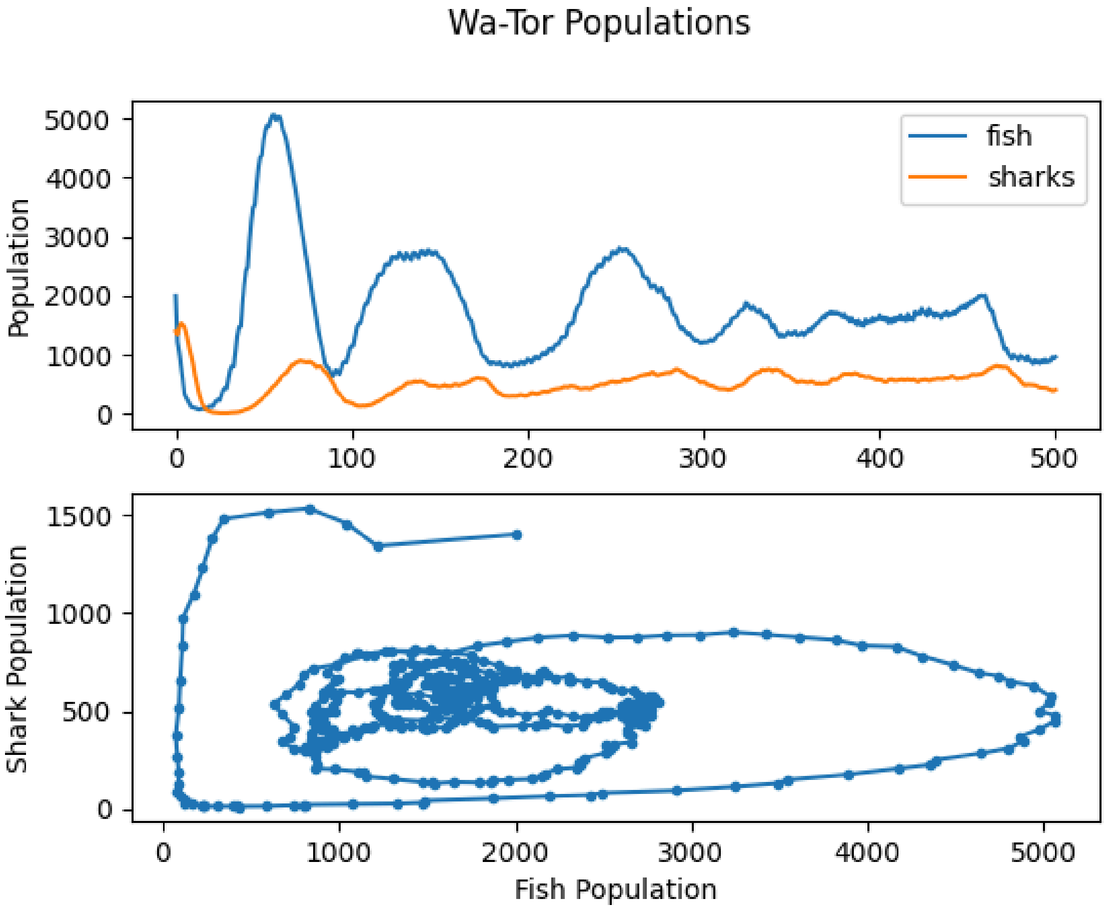


*Figure 16: Breed time: **3**, Energy gain: **3** Breed Energy: **10**, Dimensions: **100x80**, Initial Fish: **2000**, Initial Sharks: **1400**, Steps: **500**, Start Energy: **9**, Setup: **Basic***

Decreasing the energy gain by 1 seems to not result in the two other outcomes, but it seems that graph somewhat sticks to the model, but not so great compared to the previous. It seems like there is some kind of middle ground, but it seems like reducing the energy gain works better for the model than increasing it. In our case, increasing it above 5 caused it to terminate the simulation almost every time.

Finally, let's test out the breed energy. The breed energy is the number of steps a shark needs to take before breeding. We suspect that increasing this will cause the growth rate of the sharks to go down, causing the fish population to explode.

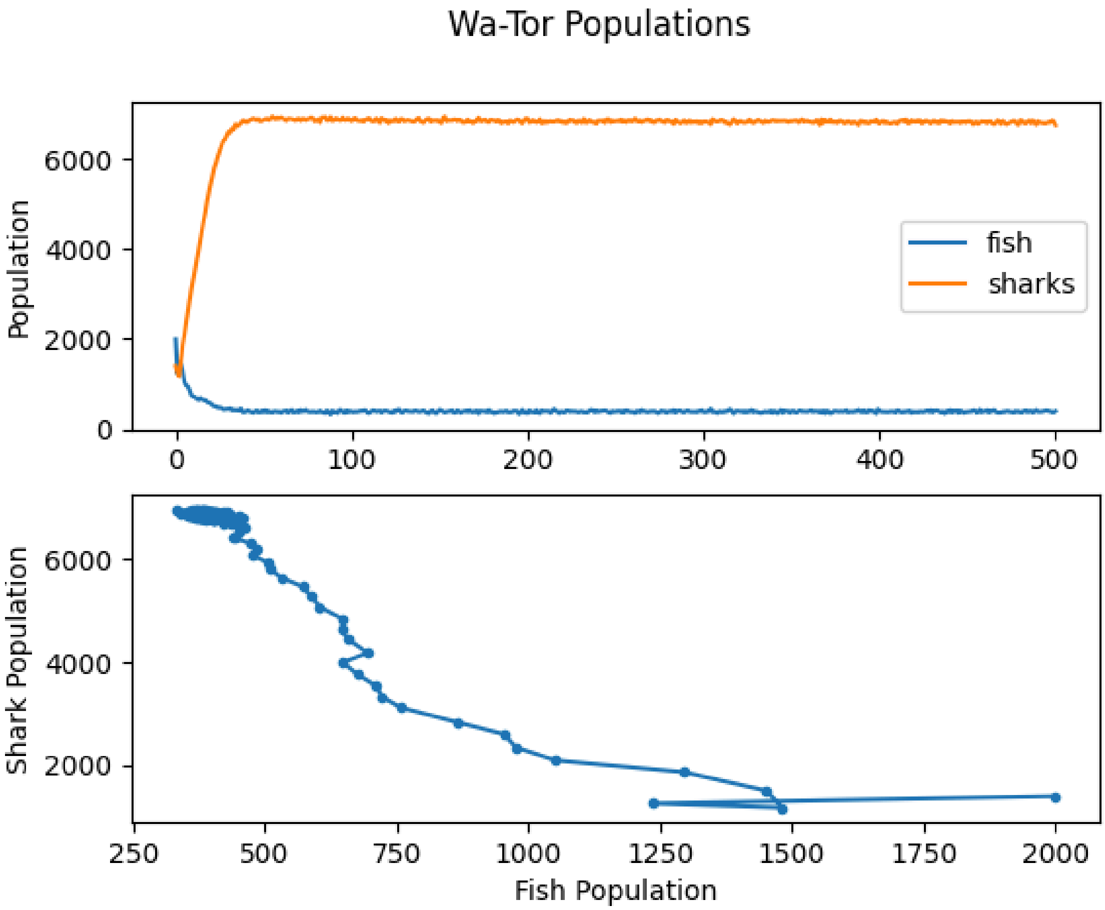


Figure 17: Breed time: ***3****, Energy gain:* ***4*** *Breed Energy:* ***5****, Dimensions:* ***100x80****, Initial Fish:* ***2000****, Initial Sharks:* ***1400****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

It seems like a lower breed energy causes the shark’s population to stay constant. This makes sense as the sharks are able to reproduce faster than they could die despite their being little to no fish.

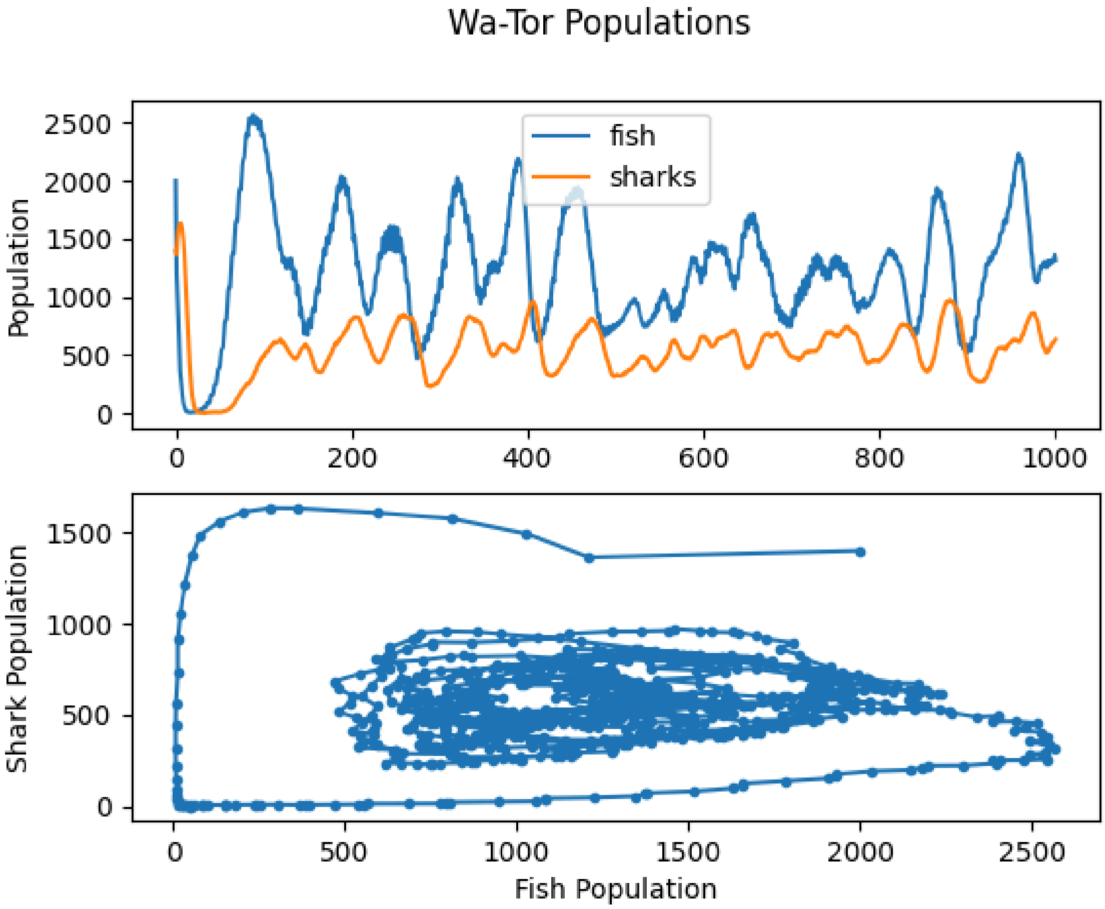


Figure 18: Breed time: ***3****, Energy gain:* ***4*** *Breed Energy: 1****5****, Dimensions:* ***100x80****, Initial Fish:* ***2000****, Initial Sharks:* ***1400****, Steps:* ***500****, Start Energy:* ***9****, Setup:* ***Basic***

As shown here in this graph, we found increasing the breed energy caused the simulation to more resemble the model compared to lowering it, but increasing still caused the simulation to end early or result in the two other outcomes more often than usual. So we suspect that this parameter is very sensitive to the type of other parameters.

In conclusion, we found that correlation is related to the population density or the total population per unit area as it affects the number of moves that a shark would need to take before reaching a fish and before dying, along with giving the fish adequate space to move around, for none of the first two outcomes listed before to occur. We also found that there was another correlation between the ratio of fish to sharks due to the fact having too much of one caused either the sharks to die too quickly or to reproduce too quickly. We also found that when altering the three main parameters, increasing the breed energy and decreasing the breed time and energy gain also proved beneficial when

Objective #3: How you think the main three simulation parameters ("breed_time", "energy_gain", and "breed_energy") should relate to the four Lotka-Volterra parameters (a, b, c, and d), and why (both theoretically and in terms of the simulation results). In order to compare to the simulation data, recall that LVM solutions oscillate around particular population values for each species that are related to the LVM parameters.

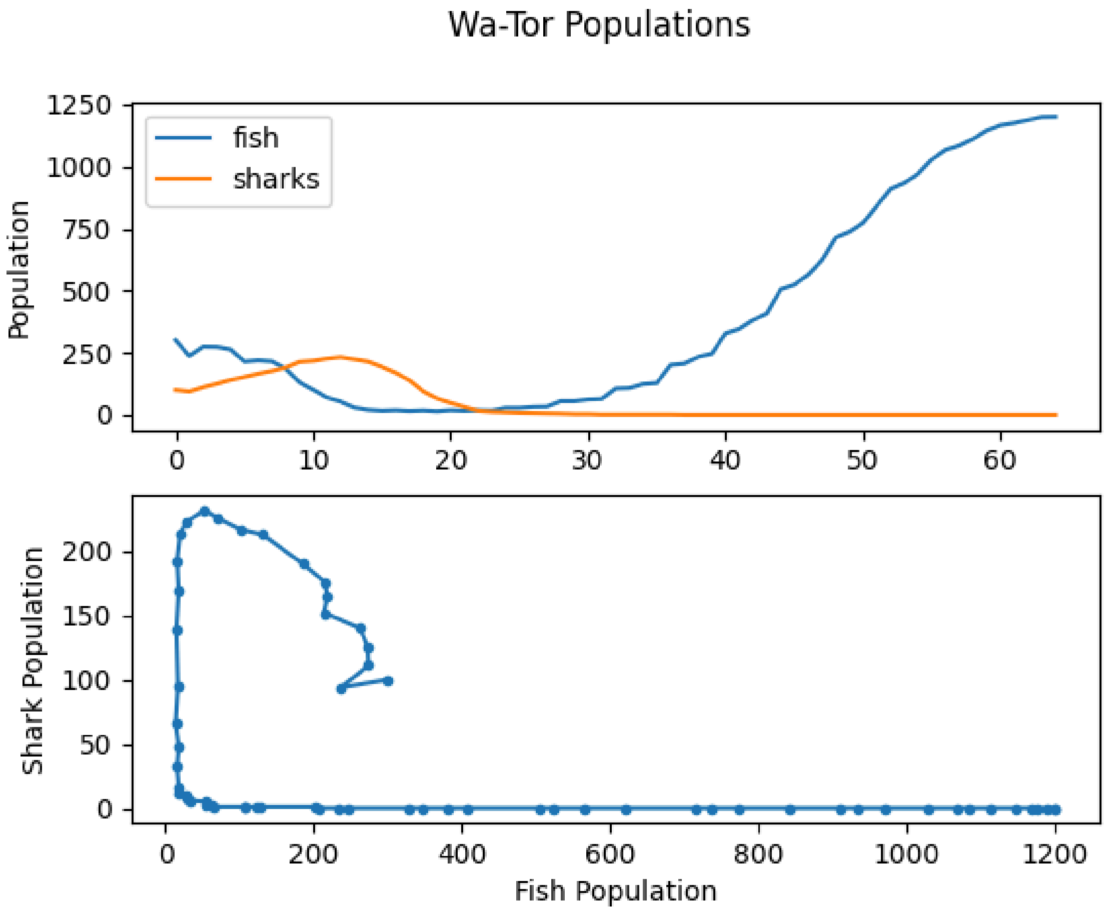


*Figure 19: Breed time: **3**, Energy gain: **4**, Breed Energy: **10***

The results in the image above can be attributed to the short reproduction period, which most probably makes the fish population grow rapidly and is in line with the high rate of prey growth, d. In other words, when the sharks get enough energy through predation, they tend to fall behind but quickly catch up. However, with the over predation drastic decline of the fish population, the predator population cannot keep up. Shark extinction follows shortly thereafter, which suggests that the predator-prey cycle is not feasible.

The Lotka-Volterra dynamics collapses because the predator is exhausting its prey too quickly, and cannot manage to recuperate itself.

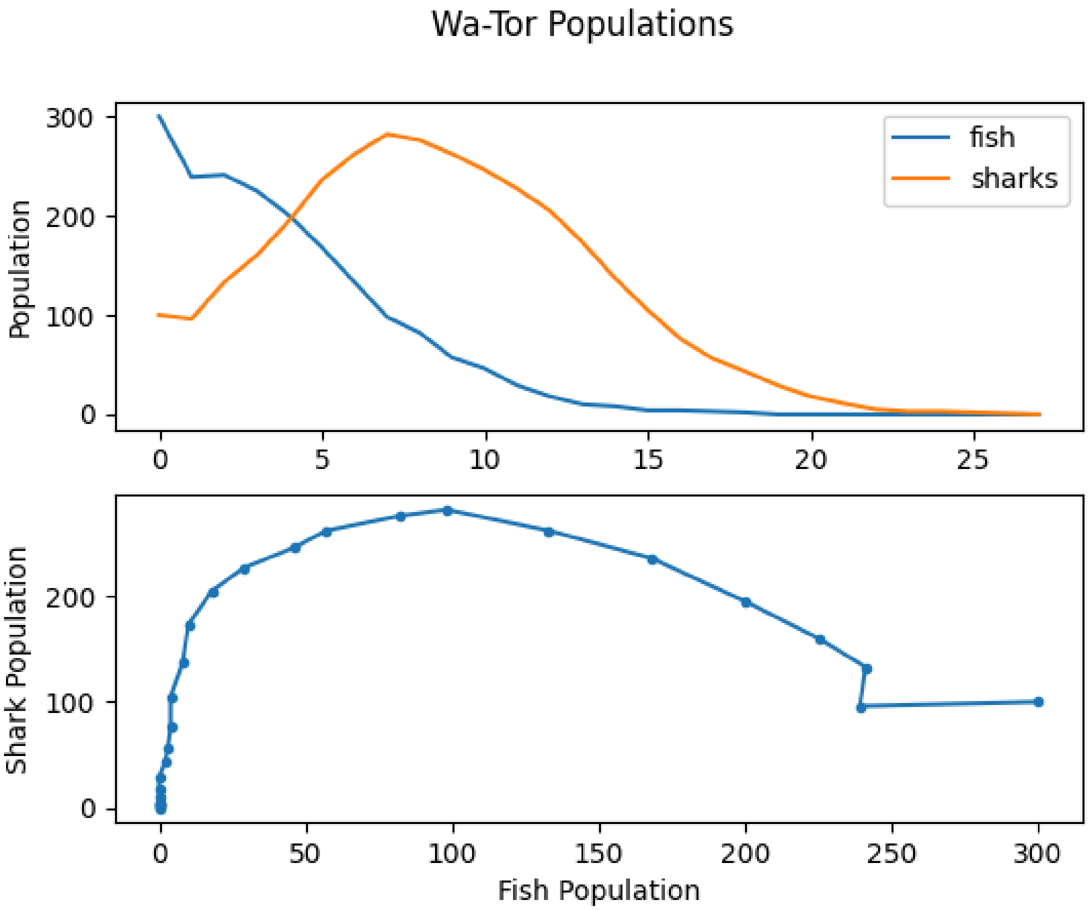


Figure 20: Breed time: ***5****, Energy gain:* ***6****, Breed Energy:* ***8***

At first for this graph, the populations of fish and sharks followed the traditional predator-prey pattern. But because of the low reproductive energy barrier, sharks breed more quickly, which increases the number of predators. This is consistent with a high growth rate (b) for predators and a moderate growth rate (d) for prey. The Lotka-Volterra cycle is broken when the shark population crashes after the fish population shrinks dramatically. This suggests that predators are outpacing prey.

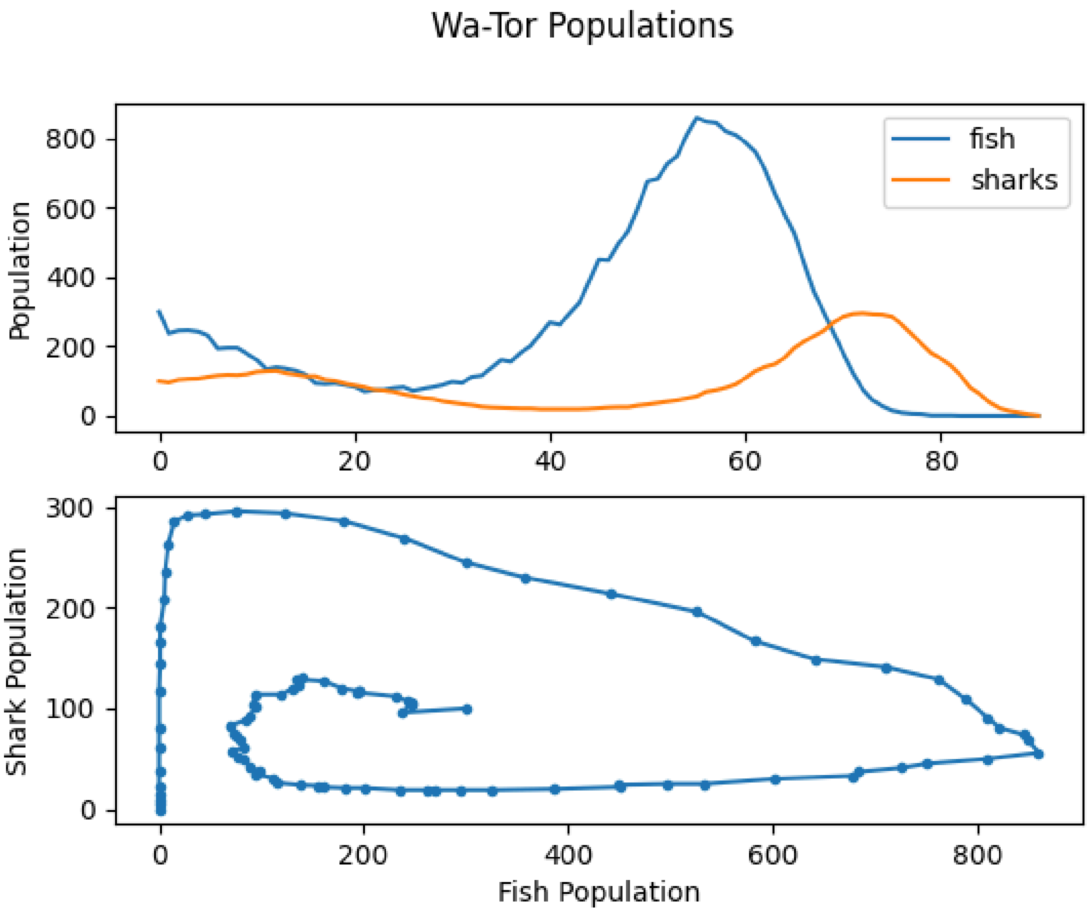


*Figure 21: Breed time: **4**, Energy gain: **3**, Breed Energy: **15***

For this graph, the population of the fish grows gradually; owing to the high energy threshold correlated with a high rate of predator death(a), sharks find procreation difficult. The predators' population never recuperates fully and always lags. This would mean that due to this, the population of sharks would go extinct eventually while the population of the fishes would recover from the initial dip. This result will confirm that, in the Lotka-Volterra model, insufficient prey can lead to the extinction of the predators that have a high decaying rate or reproduce slowly.

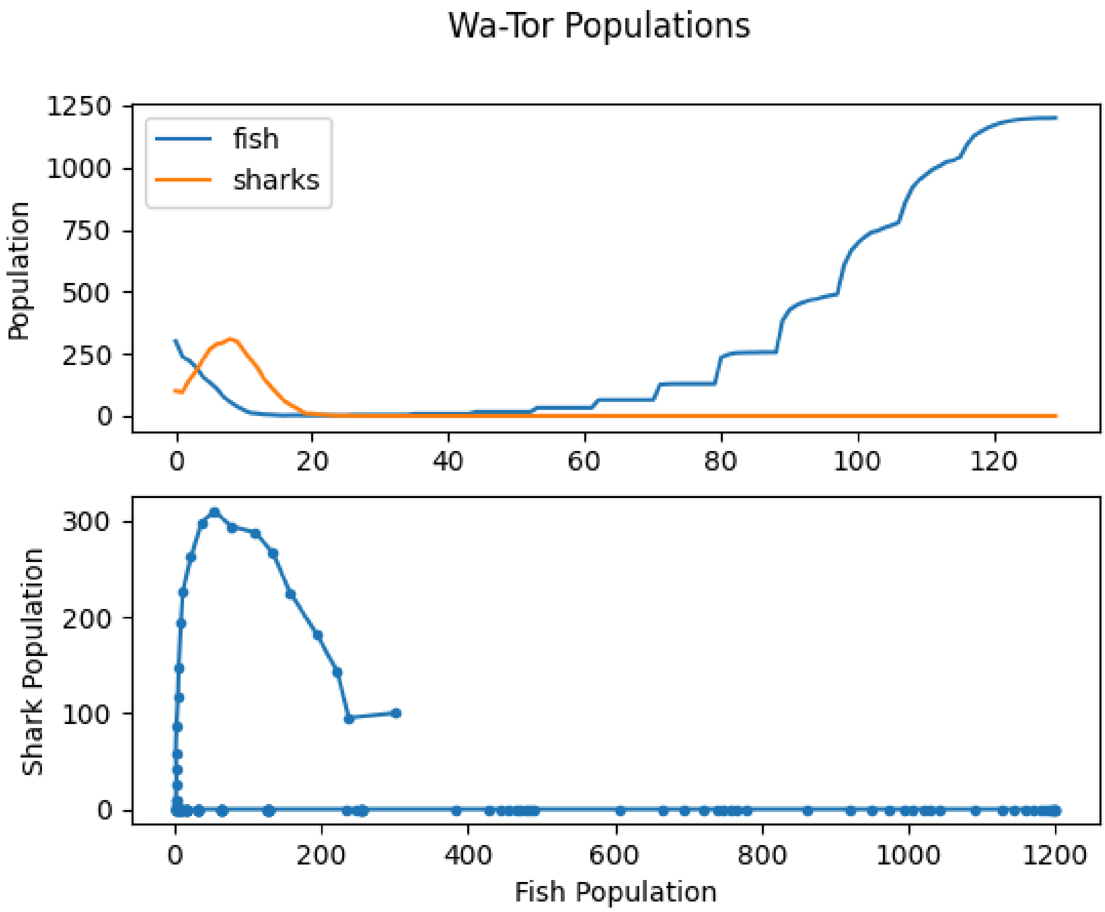


*Figure 22: Breed time: **8**, Energy gain: **7**, Breed Energy: **7***

For this graph, the shark populations receive much energy from predation, which by definition means d for the prey growth is low and that the predation rates b and c are intensive. Therefore, the population of fish is less likely to grow. As a result, shark populations declined because over-predation was the result of their initial prosperity. Eventually, they become extinct due to a lack of prey. Overly strong predation disrupts the "balance" between predator and prey and may lead to extinction. The slow recovery of prey combined with the efficiency of predators causes a break in the Lotka-Volterra cycle.

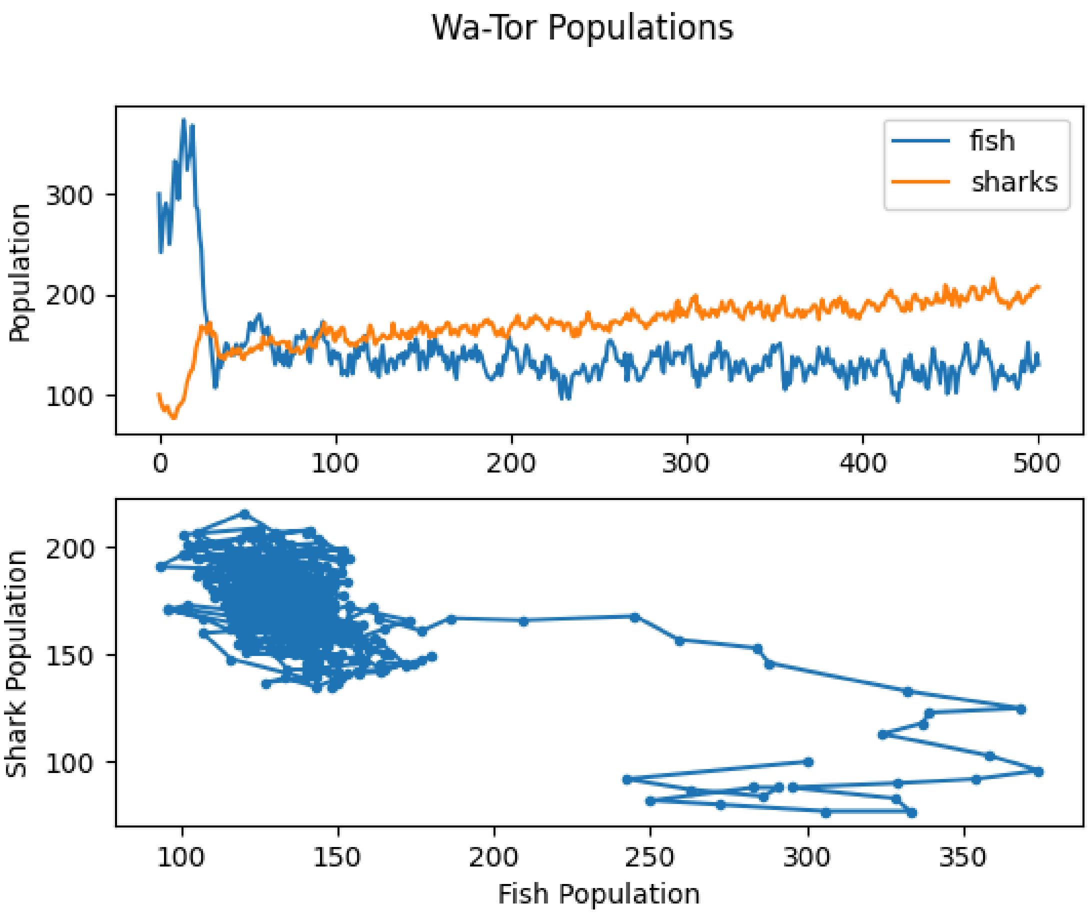


Figure 23: Breed time: ***4****, Energy gain:* ***2****, Breed Energy:* ***5***

For this next graph, both shark and fish populations fluctuate slightly, mostly around rather constant values. The low breed energy threshold for sharks enables them continuously to reproduce without exhausting the prey population, while low energy gain limits their capability to overpronate. Overall balanced dynamics, most similar to the theoretical Lotka-Volterra model, result in a more stable predator-prey interaction where both populations can oscillate sustainably about constant population numbers.

So in conclusion, the simulation results show how the three key factors of breed_time, breed_energy, and energy_gain outline predator-prey dynamics and relate to Lotka-Volterra model parameters a, b, c, and d. Many graphs show that fast prey reproduction is, small breeds leads to an initial over-supply of prey, while if the reproduction of predators becomes unbalanced, then over-predation may lead to system breakdown. If predators breed on low energy and gain energy at high rates, they can grow fast but this generally leads to the extinction of the prey and, consequently, to the extinction of the predator. On the other hand, if reproduction is too slow (high breed_energy), predator populations will not recover even when prey is abundant.

According to one of the simulations, the most stable dynamics are when the prey growth and predator efficiency balance each other out, which allows both populations to fluctuate without collapse. The Lotka-Volterra model would suggest that these results illustrate the sensitive balance that ecosystems must achieve to sustain long-term predator-prey associations.

Objective #4: The difference in the simulation results for the two types of setups the program implements (when "basicSetup" is True or False). (Not just for the default parameters)

Here is when the basic setup is True and when it is False:

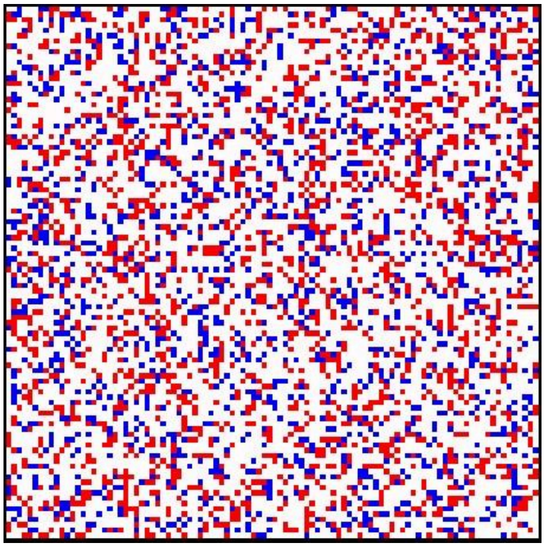


*Figure 24: basicSetup=**True** "randomness"*

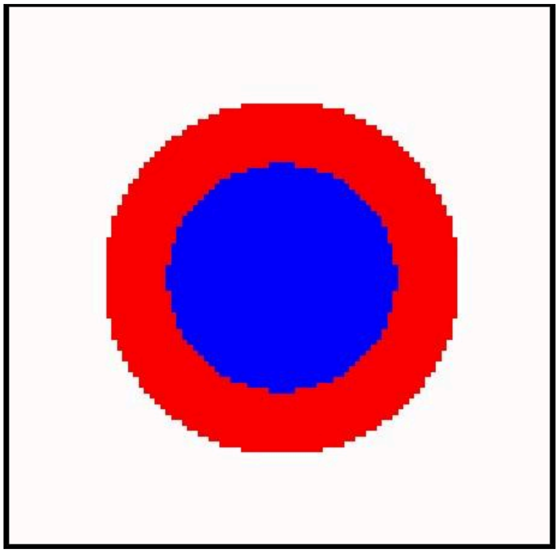


*Figure 25: basicSetup=**False** "circle pattern"*

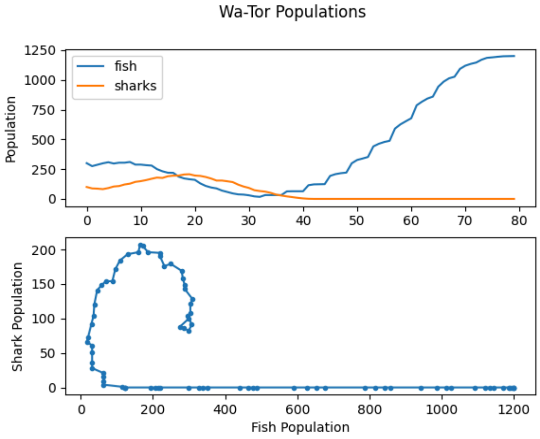


*Figure 26: Breed time: **3**, Energy gain: **4**, Breed Energy: **10** basicSetup=**False***

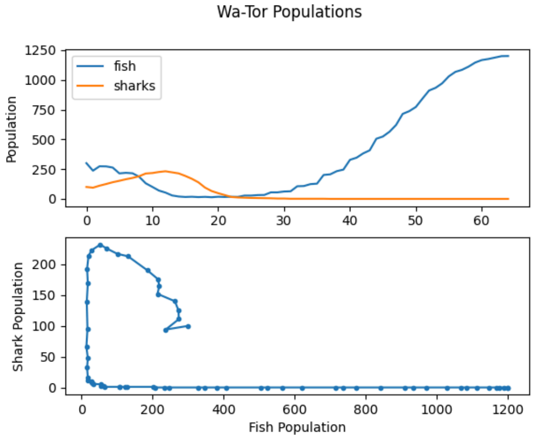


*Figure 27: Breed time: **3**, Energy gain: **4**, Breed Energy: **10** basicSetup=**True***

In the left graph above the distribution of sharks and prey is random at first. This produces a very localized variability that leads to an unequal occurrence of predation, accelerating the collapse of both species. The randomness causes an early breakdown of the predator-prey cycle, with population collapse occurring more quickly than would be expected due to area-to-area variability.

In the right graph above, this is a more balanced and therefore more predictable interaction of sharks and fish. The ultimate result is the same collapse of prey followed by predator extinction; the early dynamic can be more stable because of the structured starting distribution. The interaction is smoother due to a more uniform distribution of the populations, and the cycle before the collapse lasts a little longer.

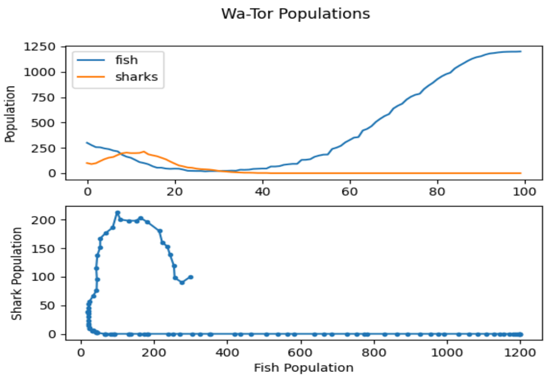


*Figure 28: Breed time: **6**, Energy gain: **6**, Energy: **8** basicSetup=**False***

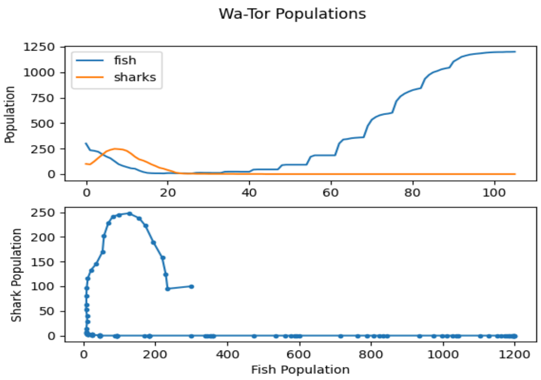


*Figure 29: Breed time: **6**, Energy gain: **6**, Breed Energy: **8** basicSetup=**True***

As the relationship between predators and prey becomes more erratic with the random initial distribution, some places start to collapse very fast, while over-predation hits some other places. The shark populations peak early and then sharply go down in some areas as prey becomes limited, thus accelerating their demise.

This organized distribution allows for more gradual dynamics of populations and thus leads to a more balanced predator-prey relationship. The cycle is longer and the populations go through smoother swings early on, but the final outcome is the same: prey collapse followed by predator extinction. The structured framework, through more balanced interactions, delays the inevitable collapse.

Extension: Alternate Setup

For our extension, we decided to implement another two setup options. Our first would be a simple tile-like pattern that starts with more fish than sharks. For our second setup, we decided to do the “polar ends” pattern, with slightly more fish on one end of the area than sharks on the other end.

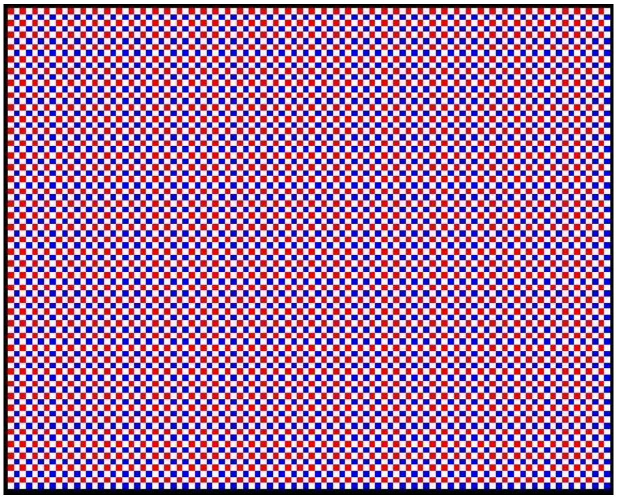


Figure 30: Tiled pattern

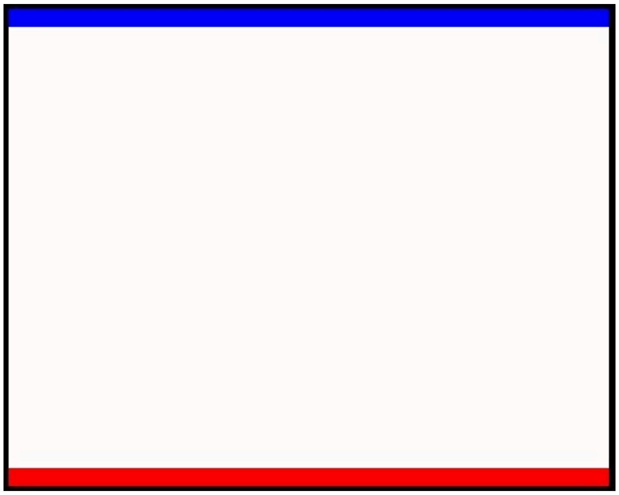
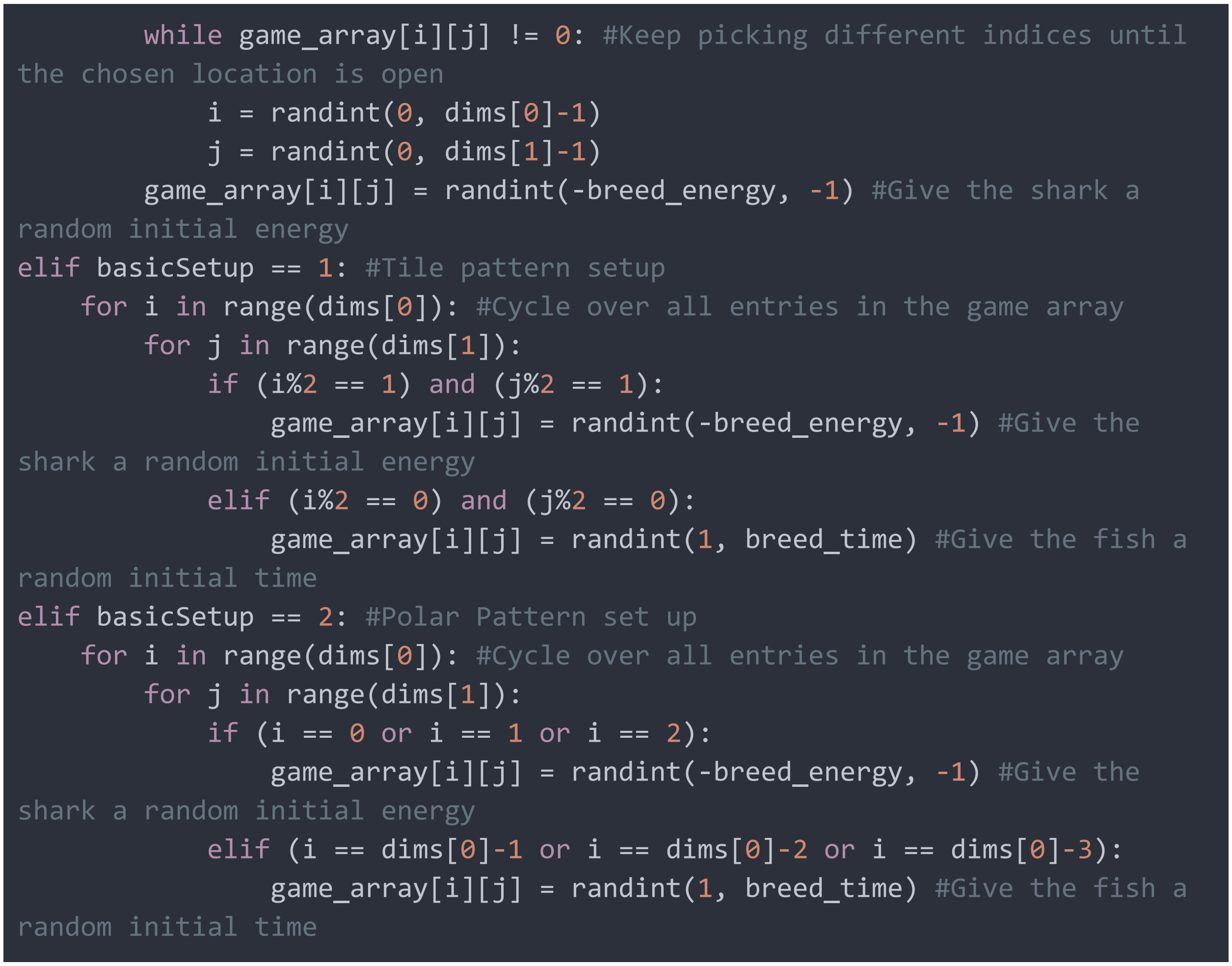


Figure 31: Polar ends

In order to achieve this, altered part of the if-else statement on line 132 of the code to include two new else-if statements that correspond with both patterns above.

Here is the altered portion of the code:

```
if basicSetup == 0: #The basic set-up with random placement of new fish and sharks
    for k in range(initial_fish): #Randomly populate the game array with new initial fish
        i = randint(0, dims[0]-1)
        j = randint(0, dims[1]-1)
        while game_array[i][j] != 0: #Keep picking different indices until the chosen location is open
            i = randint(0, dims[0]-1)
            j = randint(0, dims[1]-1)
        game_array[i][j] = randint(1, breed_time) #Give the fish a random initial time
    for k in range(initial_sharks): #Randomly populate the game array with new initial sharks
        i = randint(0, dims[0]-1)
        j = randint(0, dims[1]-1)
```



```
        while game_array[i][j] != 0: #Keep picking different indices until the chosen location is open
            i = randint(0, dims[0]-1)
            j = randint(0, dims[1]-1)
        game_array[i][j] = randint(-breed_energy, -1) #Give the shark a random initial energy
elif basicSetup == 1: #Tile pattern setup
    for i in range(dims[0]): #Cycle over all entries in the game array
        for j in range(dims[1]):
            if (i%2 == 1) and (j%2 == 1):
                game_array[i][j] = randint(-breed_energy, -1) #Give the shark a random initial energy
            elif (i%2 == 0) and (j%2 == 0):
                game_array[i][j] = randint(1, breed_time) #Give the fish a random initial time
elif basicSetup == 2: #Polar Pattern set up
    for i in range(dims[0]): #Cycle over all entries in the game array
        for j in range(dims[1]):
            if (i == 0 or i == 1 or i == 2):
                game_array[i][j] = randint(-breed_energy, -1) #Give the shark a random initial energy
            elif (i == dims[0]-1 or i == dims[0]-2 or i == dims[0]-3):
                game_array[i][j] = randint(1, breed_time) #Give the fish a random initial time
```

Figure 33: Modified code

When approaching this, we decided to keep the parameters the same for both tests in order to get consistent results

Our setup parameters goes as follow:

Breed time: **3**

Energy gain: **4**

Breed Energy: **10**

Dimensions: **80x100**

Steps: **500**

Start Energy: **9**

First, start with the tiled pattern. We ended up getting either two of the outcomes shown below.

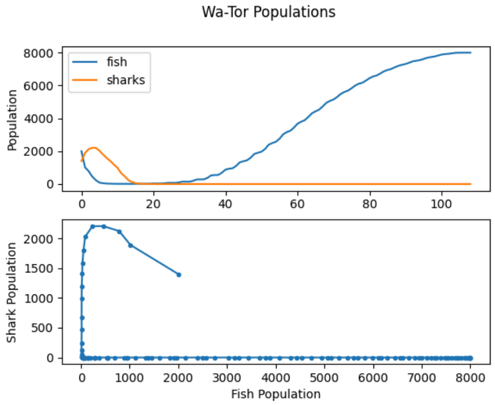


Figure 34: Result 1

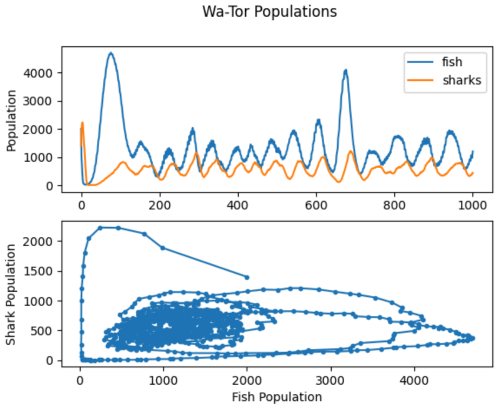


Figure 35: Result 2

The simulation ended with either the population resembling the LVM or with just the sharks and fish living. But overall, we found that the setup tended to just end up as the fish outliving the sharks, so this setup doesn't match the LVM most of the time. This was most likely due to fish and sharks being so close together, it caused the fish to be eaten too quickly or

Next, let's test out the other setup, the polar ends. This setup includes a somewhat equal amount of sharks and fish on the opposite ends of the area. During testing, we found the setup nearly matched the LVM most, if not all the time.

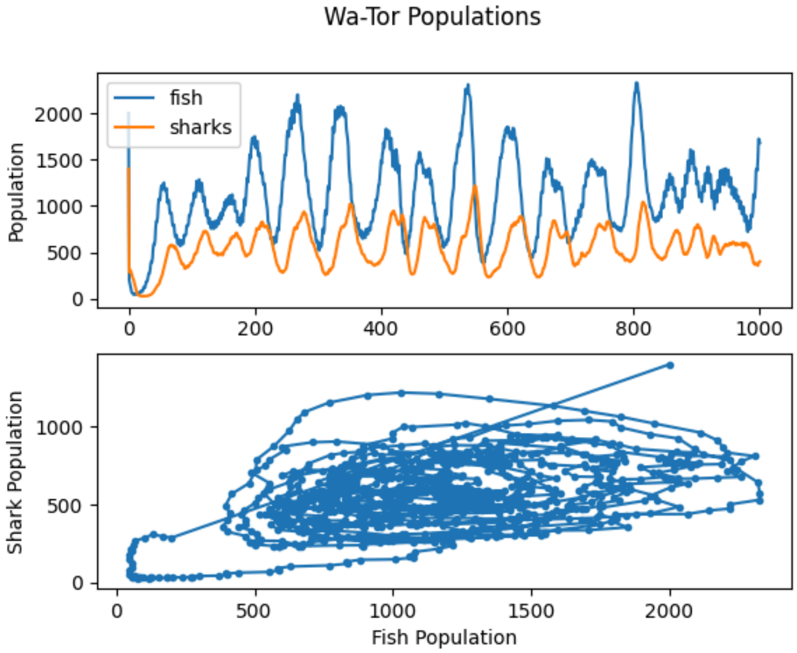


Figure 36: Only Result

As shown in the graph above, we found this setup to be quite ideal in at least resembling the LVM. The graphs of course the sharp drop in population and then both populations hitting equilibrium after resembles the LVM.