Predator-Prey Model

Connor Hann, Xiaomeng Jia, Peifan Liu and Xinyu Wu[1]

[1]*Physics Department, Duke University*

(Dated: April 7, 2016)

Abstract

To be added.

BACKGROUND

Modeling the interactions between predator and prey is a question of great ecological importance, as accurate models can allow one to make reliable predictions and thereby inform decisions in conservation, habitat preservation, hunting regulations, etc. There exist a variety of methods for modeling the interactions between predator and prey, and these models, though simple, are often in quite good agreement with what is observed. Take the classic Lotka-Volterra Model as an example. While this model is nothing more than a set of first-order nonlinear differential equations that one may solve numerically for a particular set of initial conditions, the model is capable of providing a qualitatively accurate description of the data, as is shown in Fig. 1.

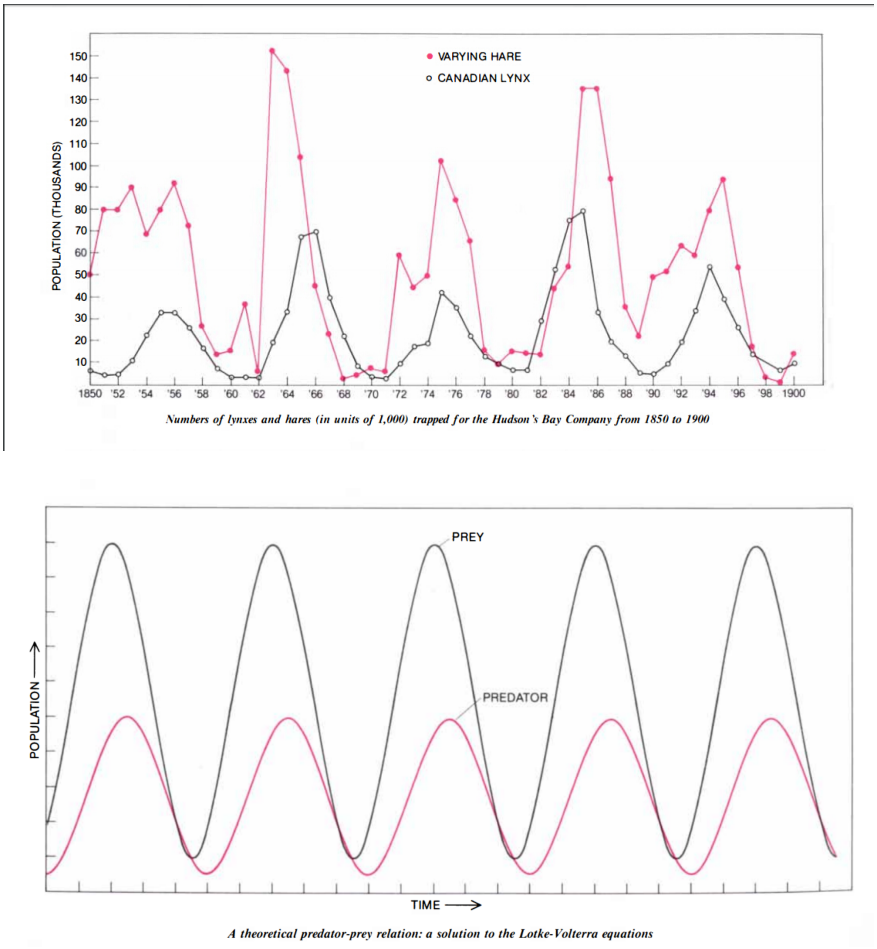

FIG. 1: (Top) experimental data of Hare and Lynx populations, (Bottom) predictions from the Lotka-Volterra equations.

In this work we study the behavior of a slightly different model, one that is similar to the Lotka-Volterra Model in that it consists of only a simple set of rules, but the nature of the actual simulation is quite different. In our case, we consider two populations, sharks and fish living on a two-dimensional lattice. We simulate the movement and behavior of each shark and fish individually and can observe fluctuations of the two populations as a function of time. Unlike the Lotka-Volterra Model, however, we are able to see exactly where all of the sharks and fish are and this allows us to attain a deeper understanding of what causes the fluctuations in populations and the conditions that may lead to extinction.

IMPLEMENTATION

The model we consider is remarkably simple in that there are only five user input parameters: the initial number of sharks $n0\_sharks$, initial number of fish $n0\_fish$, the time it takes fish to breed $breed\_age\_Fish$, the time it takes sharks to breed $breed\_age\_Shark$, and the time it takes a shark to starve $starve\_time$. To begin the simulation, one creates $n0\_sharks$ sharks and $n0\_fish$ fish at random locations on a two-dimensional lattice, and each fish or shark is given a random age between $breed\_age\_Fish$ and $breed\_age\_Shark$, respectively. At each subsequent time step of the simulation both sharks and fish will swim around the grid, and the sharks will attempt to eat the fish. Below we discuss the specific rules that dictate the movements of the fish and sharks.

Fish Movement

During a time step, each fish looks at the four adjacent lattice sites and chooses an empty site to which to move. If no lattice sites are available, the fish remains in its current position. We implement periodic boundary conditions so that if a fish moves through one boundary it reappears on the boundary on the opposite side, i.e. our two-dimensional grid is actually a topological torus. The age of the fish is also increased by one. If the age of the fish is $breed\_age\_Fish$, an additional fish is created at the site the fish has just moved from, and the ages of both fish are set to 0.

Shark Movement

The goal of the sharks is to eat fish, so during a time step each shark will look for fish on the adjacent lattice sites. If fish are found, the shark chooses one randomly and moves to the fish's position on the lattice, deleting and "eating" the fish. If no fish are available the sharks move just like the fish, choosing an available lattice site to move to randomly. Like the fish, the age of the sharks is increased by one each time step, and sharks breed in the same way as the fish once they have reached $breed\_age\_Shark$. Unlike the fish, however, the sharks will starve if they do not find food. An array keeps track of how long it has been since each shark has last eaten a fish, and if a shark has not found a fish within $starve\_time$ time steps, it "starves" and is deleted from the simulation.

Program Structure

We use a set of five arrays to keep track of the information necessary to run the simulation as described above. Two $N \times N$ arrays, $Sharks$ and $Fish$, are used to keep track of the positions of the sharks and fish respectively. If the (i, j) site is empty the arrays hold the value -1, otherwise the site holds the age of the shark or fish. Two additional arrays $Sharkmove$ and $Fishmove$ are used to store the new positions of the sharks and fish as an update is being performed, allowing us to ensure that each shark or fish is moved only once and that two animals are not moved onto the same lattice site. Finally, a fifth array $Sharkstarve$ stores the amount of time since each shark has last eaten. The arrays are updated, first fish then sharks, via the rules above. One can plot the positions of the sharks and fish at any time step, and an example plot is shown in Fig. 2.

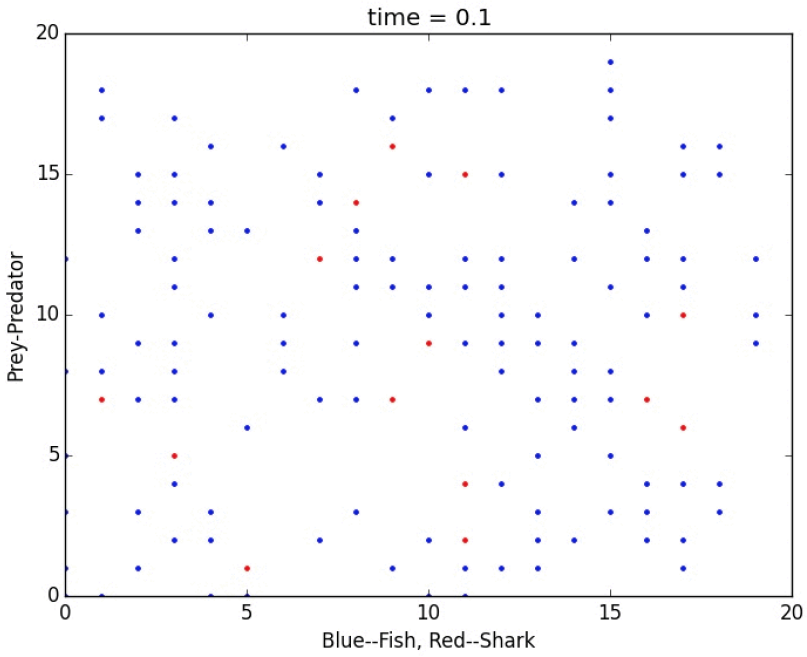

FIG. 2: A typical configuration of sharks and fish on a 20×20 lattice

NUMERICAL RESULTS

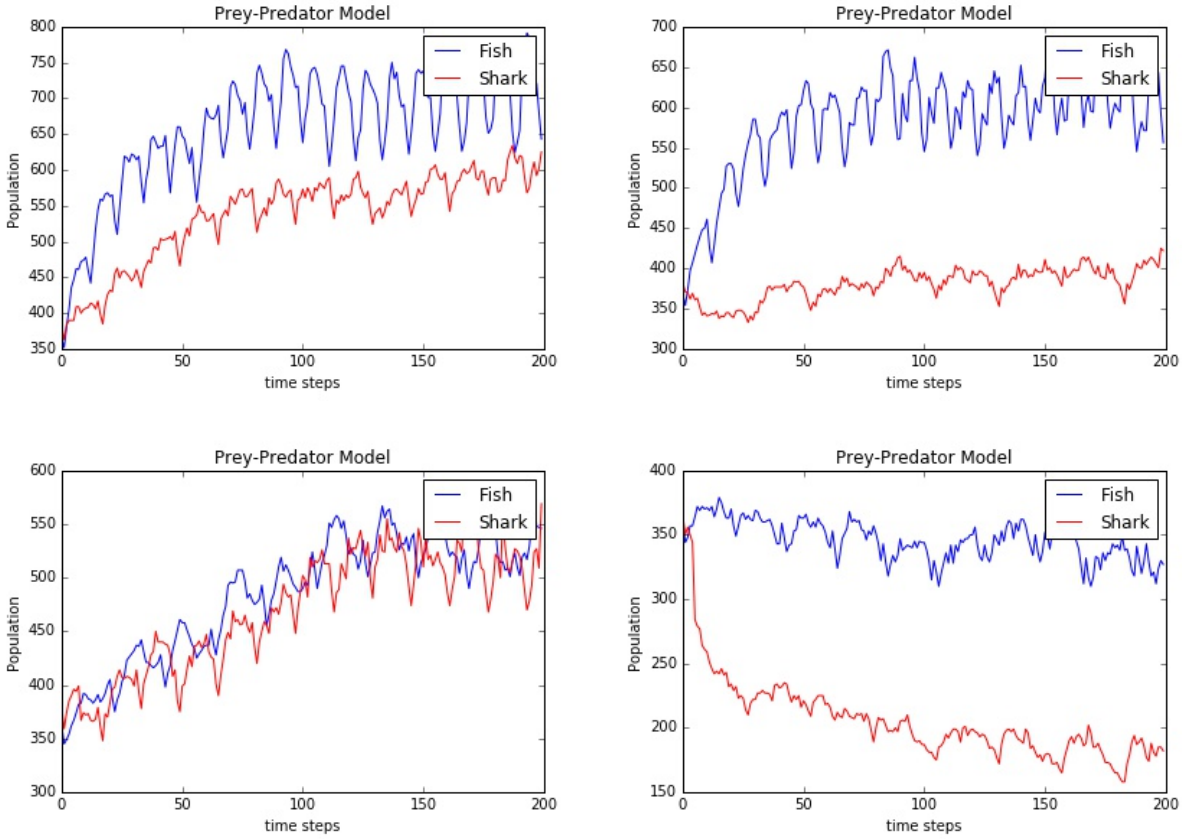

FIG. 3:

CONCLUSION