Ronney Aovida

12/12/16

ISC4232

Dr.Quiafe

Predator vs. Prey: Using Cellular Automation Modeling and Euler’s Method

**Background:**

The basis of this project is inspired by the act of Predators vs Prey around the world and how they are able to coexist in the same area without either animal ever going extinct. To figure out how earth is able to keep such a stable ecosystem that can strive off itself in the cycle of life, we as scientists had to conduct experiments and observe/analyze the results in order to obtain answers. Creating an experiment however with any controlled variables was not easily done. since creating an environment that had a controlled ecosystem was not easily feasible since we have limited knowledge about how animals will react to an environment. It is to the ecology worlds surprise when Isle Royale was founded in the 1940s. This piece of land isolated by the frigid waters of Lake Superior is home to a moderate size of wolves and moose. Due to the fact that moose and wolves are natural predator and prey and that the land they coexist on is isolated from any outside influence the scientists found this to be a perfect opportunity to observe and analyze the ecosystem over the years and record the amount of moose and wolves over time. With moose having a large amount of nearly limitless food supply provided by the land itself they were able to survive in Isle Royale in good condition, however if these moose continued this trend of living off the land and breeding for many years in peace, eventually the land itself would not be able to contain such a large population of moose, this is where the introduction of the predator comes along. In the 1930s Isle Royale was founded by humans and moose, where the humans did nothing but observe and the moose bread without conflict, however this all changed in the 1940s where a few wolves wandered over an ice bridge through the water and found themselves in a land full of potential food. The wolves feasted upon the moose for many decades and would only through starvation or natural death, for it was rare for a wolf to die during a hunt since most wolves move in packs and moose tend not to be very social creatures. In result of this the wolves became very dependent on their prey, for whenever the population of the moose dropped, so would the population of the wolves. since the moose population eventually became very scarce throughout and the wolves had a hard time finding any food, and in result would die of hunger. Over time, the wolf population decreased and the moose population increased since the number of predators on the island was so low. Now this rise in moose should not be occurring if both the animals are breeding at the same rate, but this is not the case. In response to its environment, animals have adapted their breeding customs to be the most optimal for survival. If an animal has no problem acquiring food for survival, they are capable of supporting their whole species no matter the size. However, if the animals cannot acquire food with ease, they cannot afford to be giving birth so many offspring since they cannot support them and themselves as well. It is because of this that predators around the world tend to breed at a much lower rate than most prey do in order to survive. The trend of Predators and Prey both rising and falling in population went on for decades as the observers recorded data, only to find the only factors that influenced this trend were outside factors such dramatic temperature changes, disease or the crossing of even more wolves over the ice bridge.

**The implemented Math**:

The mathematical model used is called, Lotka-Volterra.



The above equations are relatively simple, they are used to find the derivation of how much the prey population fluctuates and how much the predator population fluctuates. The formula involving dx/dt represents the change in prey whereas the formula involving dy/dt represents the change in predators. In both formulas the variables contain the same meaning, where x is the number of prey, y is the number of predators, t represents time and the Greek symbols (ex.alpha/beta) are all positive real parameters describing the interaction of the two species. In this simulation however we will be representing these parameters as the following: alpha being the reproduction rate of prey and beta being the mortality rate of predator per prey. In the predator equation, delta will be represented by the reproduction rate of predator per prey, and gamma being the mortality rate of predators.

Through these equations, we are able to predict the population of a set of predators vs. prey as long as the following conditions are taken into account:

* The prey population finds ample food at all times.
* The food supply of the predator population depends entirely on the size of the prey population.
* The rate of change of population is proportional to its size.
* During the process, the environment does not change in favor of one species and genetic adaptation is inconsequential.
* Predators have limitless appetite.

The Cellular automation model will simulate the interaction between predator and prey in a real- time simulation, hence not putting the lotka voltera equation to use. In order to see if the differential equation is an accurate model for measurement, we will use Euler’s method. Euler’s method is a first order time stepping method that calculates the next time step for a given differential equation given an initial value. Euler’s method is y(n+1)=y(n)+dt\*f(y(n),x(n)) where for example: y= prey , x=predator, n=time step, f(x)= differential equation and dt= step size. It is important to note that the larger the step size, the less accurate the results will be leading to the final result. Below is the Matlab code used to determine the step sizes.

fxwolf=@(w,m) D\*(w\*m)-G\*w;

fxmoose=@(w,m) A\*m - B\*(m\*w);

for i=1:2500

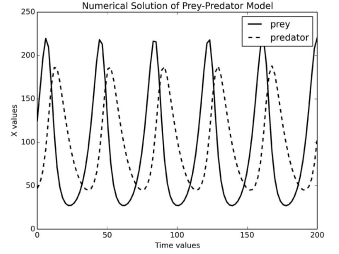
time(i+1)=i;

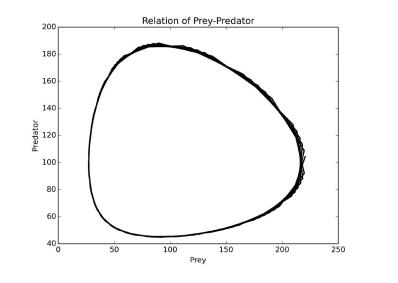
m(i+1)=m(i)+dt\*fxmoose(w(i),m(i));

w(i+1)=w(i)+dt\*fxwolf(w(i),m(i));

end

Lotka-Volterra graphical representation: (first graph is figure 19.12, second is 19.13)

* “Figure 19.12 shows the graph of the population changes of the prey and predator over time. After 10 seconds, the displacement changes linearly. Figure 19.13 shows the phase plot of the two population changes. Because this a closed curve it implies that the prey and predator populations follow periodic cycles”- cited from: Introduction to Computational Models with Python by Jose Garrido
* 



**The program and how it works:**

The program will take in the parameters asked by the main function (see below) and will process these parameters through a grid size determined by the user for a certain amount of “ticks”. These “ticks” are how the program is capable of producing a real- life simulation of the predators vs. prey. For every “tick” that occurs, the animals on the grid will use the check grid method to check the availability of every square coordinate around its current location (8 possible surrounding coordinates) and will have the animal “do” something. The things the animals are capable of doing every “tick” are: eating nearby prey, breeding if its time, move if not eating or moved already and die if starved or eaten. The “ticks” are also responsible for determining when it is time for a animal to breed or starve by decrementing the set “ticks” given by the user for the animals breed/starve time by one for every tick the program runs through. The tick timer for starvation however will reset every time a predator eats a prey. However, when it is time to breed, an animal will search the spots surrounding it and initialize a new animal in an available spot. If there are no available spots for breeding, the ticking for that method will not go through until the next tick of the program, the breed timer will not simply reset if this predicament occurs.

The program will continue to have the predators and prey move around the grid at random locations set by the check\_grid method having the animals eat, breed and die all over the grid every tick until one of the populations go extinct or the amount of ticks run out.

It consists of 4 classes:

Each Class incorporated the following methods (took in the following parameters):

* Class Island(object)
  + \_\_init\_\_(self,n,prey\_count=0,predator\_count=0)
    - This method takes in an object, the size n of the grid of size nxn, and the amount of predator/prey. To create a grid, it creates a list/vector of n rows of all zeros and calls the init\_animals method to initialize the amount of prey\_count and predator\_count animals to the grid.
  + init\_animals(self,prey\_count,predator\_count)
    - This method takes in an object, and the amount of prey and predators. It uses random.sample() to create a list of random numbers 0:n of size n-1 and finds an x coordinate by taking a number from the list and dividing it by n and rounding down, find an y coordinate by taking a number from the list and finding its remainder divided by n. it will then put them together to create a (x,y) coordinate to place a prey or predator on depending on how many of those animals are called upon.
  + clear\_all\_moved\_flags(self)
    - animals that have a moved flag that indicated they moved during the last tick have their flag removed to they are clear to move the next tick.
  + Size(self)
    - Returns the size of the island: one dimension.
  + register(self,animal)
    - registers an animal with the island and places said animal at its coordinates using the animal method
  + remove(self,animal)
    - removes an animal from the island using the animal method
  + animal(self,x,y)
    - returns an animal at location (x,y) as long as x and y are between 0 and the grid size, otherwise it is deemed outside the island boundary being givena value of -1
  + \_\_str\_\_(self)
    - prints out a visual grid containing “.” For empty spaces and “X”/”0” for predator/prey spaces respectively (the “X” AND “0” are default signs given to the animals and can be changed by the user)
  + count\_prey(self)
    - counts all the prey on the island and returns that number
  + count\_predators(self)
    - counts all the predators on the island and returns that number
* Class Animal(object)
  + \_\_init\_\_(self,island,x=0,y=0,s=”A”)
    - initializes the animals and their positions by taking information from the island class. An “A” is printed to the grid if it is not assigned as a predator or prey.
  + position(self)
    - returns coordinates of the animals current position
  + \_\_str\_\_(self)
    - prints out the value given by self.name
  + check\_grid(self,type\_looking\_for=int)
    - looks in the 8 directions from the animals location and returns the first location that presently has an object of the specified type. And returns 0 if no such location exists.
  + move(self)
    - moves the animal to a nearby position and assigns that animal a flag dictating it has moved that turn, as long as check\_grid returns true
  + breed(self)
    - breeds a new animal and if there is room in one of the 8 locations nearby it will place a new prey there, otherwise it must wait until a spot is open.
  + clear\_moved\_flag(self)
    - this will set all the moved flags to be false, hence removing them for that tick
* Class prey(Animal)
  + \_\_init\_\_(self,island,x=0,y=0,s=”0”)
    - takes in the Animal class, initializes the preys breed clock and sets the print value for prey as “0”
  + clock\_tick(self)
    - this will decrement the **local** breed clock for prey
* Class Predator(Animal)
  + \_\_init\_\_(self,island,x=0,y=0,s=”X”)
    - takes in the Animal class and initializes the starve and breed clocks. Assigns the print value for predators for be “X”
  + clock\_tick(self)
    - this will decrement the **local** breed and starve clock for predators. If the animals starve clock reaches 0 however, it will use the remove method to remove the animal.
  + eat(self)
    - the predator looks for one of 8 nearby locations for prey, if found it will move to that location and reset its starve clock and remove that prey from the grid.

**The main function:**

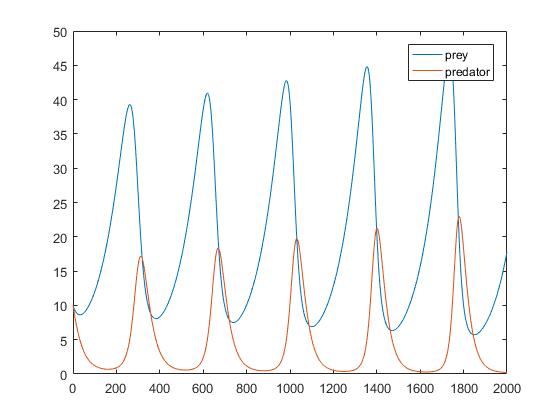
* Parameters main takes in:
  + (predator\_breed\_time=xx,predator\_starve\_time=xx, initial\_predators=xx, prey\_breed\_time=xx, initial\_prey=xx, size=xx,ticks=xx)
    - xx=user input
  + the main function is known as the “main simulation”, the user sets the parameters and receives an output of event loops every 10 ticks displaying a grid of empty spaces, predators and prey. Also displays a plot of the population of both predators and prey over time until one of them went extinct, or until max ticks.

**Results/Observations:**

To find the most ideal paramaters for the Cellular Automation Model and Euler’s Method, the programs are run multiple times at different parameters, until the output plot represented something similar to the lotka voltera graph seen above (the implemented math section).Once these Ideal parameters are found it is then changed one variable at a time to see what happens when one simple change occurs in this delicate system.

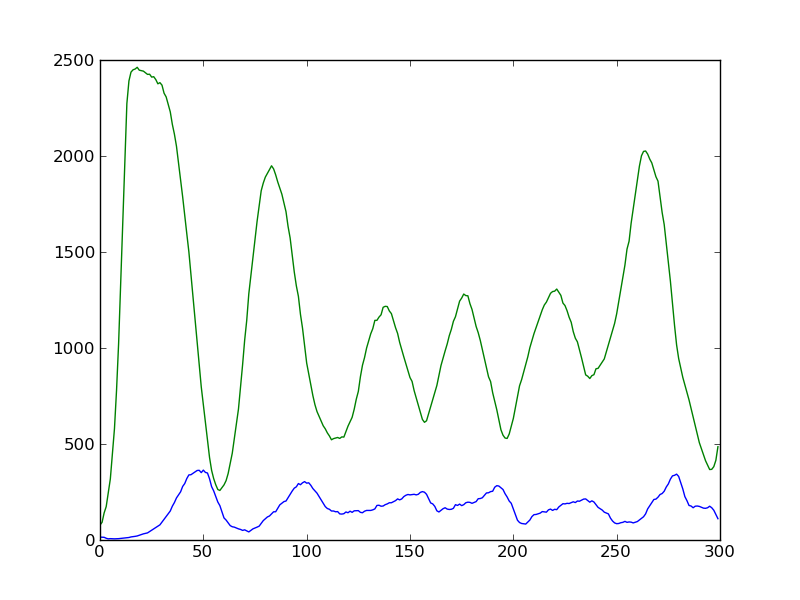
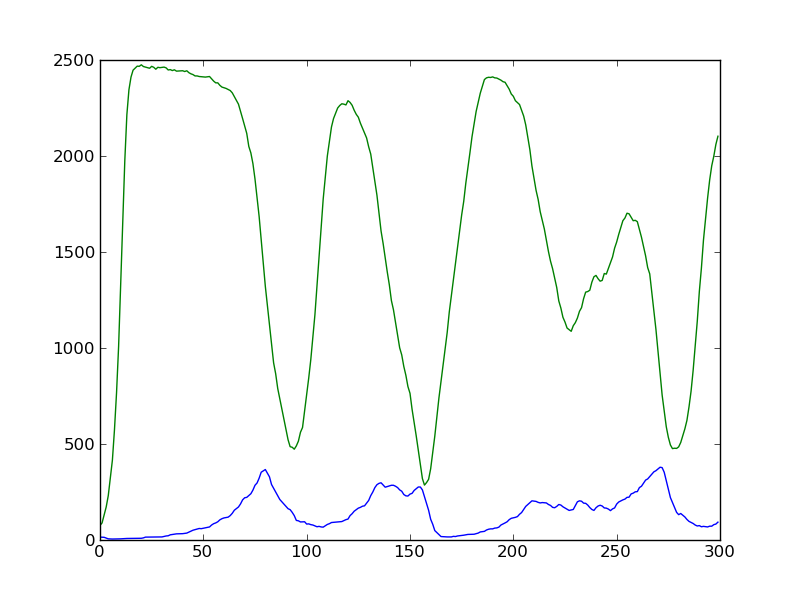
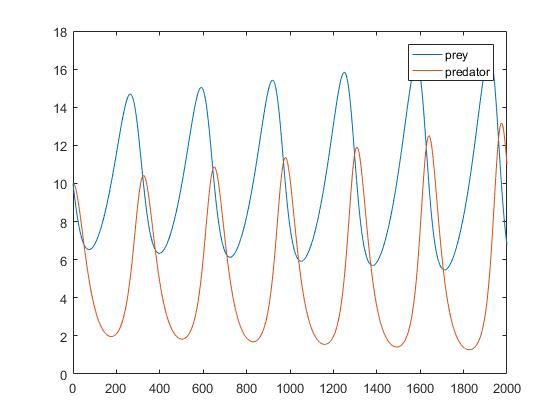
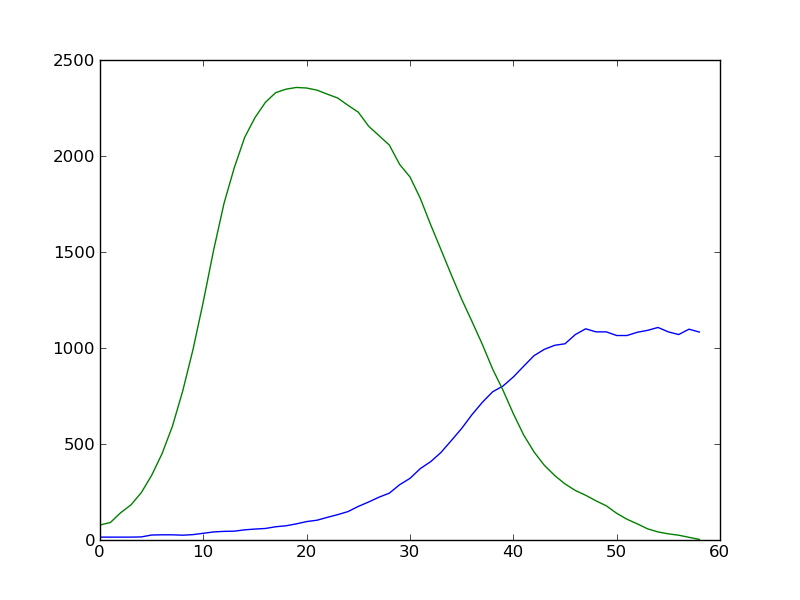
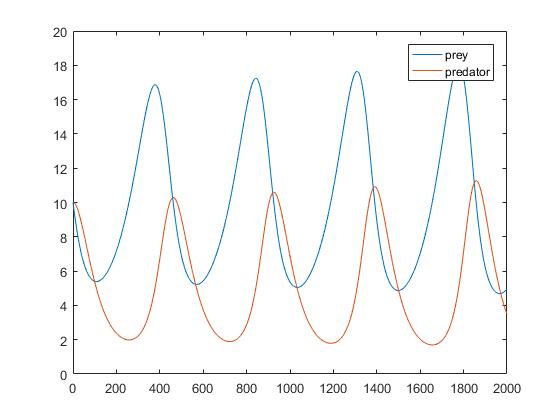
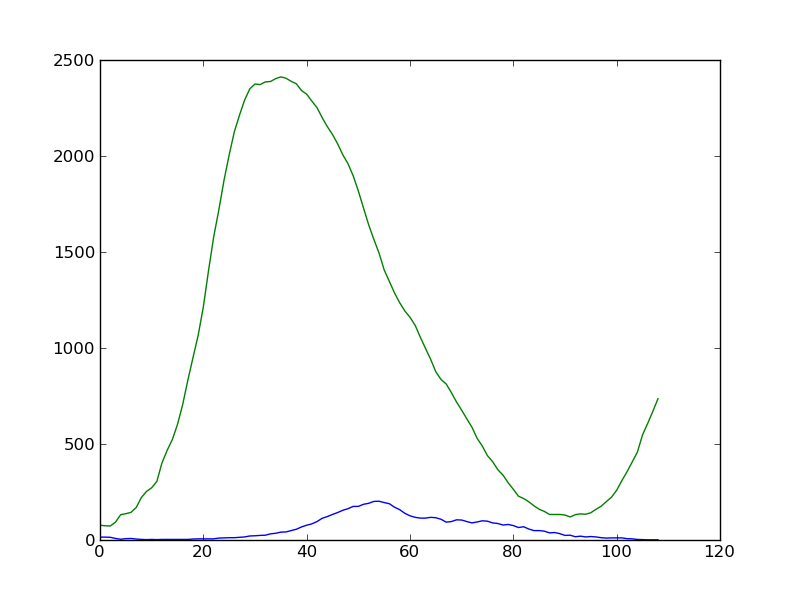
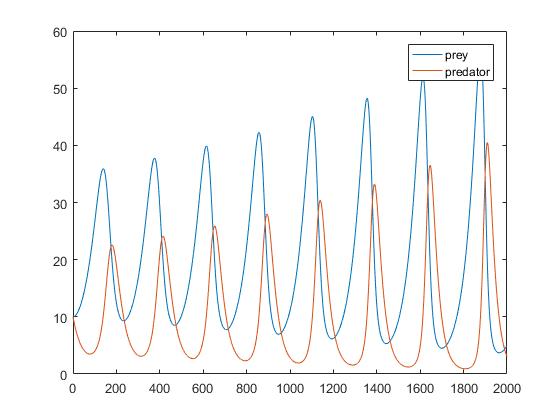
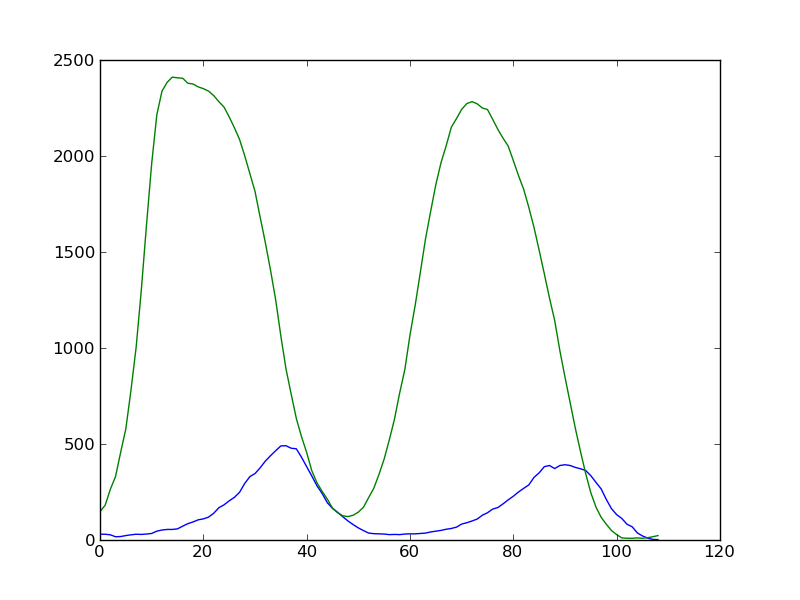
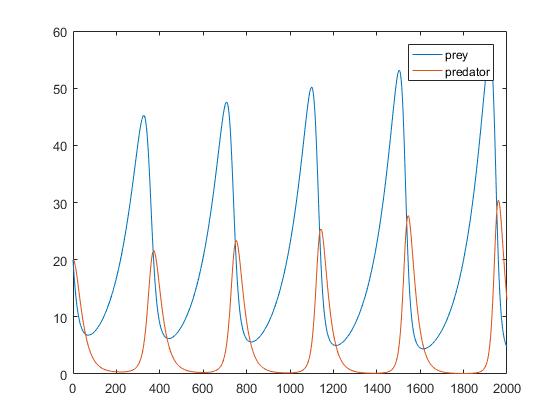
* The most ideal parameters for Euler’s Method:
  + reproduction rate of prey=0.1
  + mortality rate of predator per prey= .02
  + mortality rate of predator= 0.4
  + reproduction rate of predator per prey= .02
  + time step = 0.1
  + initial moose =10
  + initial wolves =10
  + In this run the predators and prey are able to coexist for an extremely large amount of time with many increases and decreases happening

at the same time since predator population depend on the prey.

* + 
* The most ideal parameters for the Cellular Automation Model:
  + Predator: breed time= 8, starve time= 6, initial predators= 15
  + Prey: breed time= 3, initial prey: 80
  + 40x40 grid size and 300 ticks
  + In this run the predators and prey are able to coexist for an extremely large amount of time with many increases and decreases happening

at the same time since predator population depend on the prey.

(green=prey, blue=predators)

* + 
* Predators breed time is doubled
  + CAM(cellular automation model):In this run, you can see the moose and wolves are able to go longer periods of having a larger or smaller population since the predators breed time takes so long
  + 
  + Euler’s Method: The population of predators never dips below 2 or 3 since their reproduction rate is increased, but also limits the growth potential of both animals since the prey cant reproduce fast enough to keep up with the predator population.
  + 
* Predators take a longer time to starve
  + CAM:In this run the predators take a much longer time to die from starvation, and in result grow in numbers so large that they out- number the prey and drive their food source to extinction.
  + 
  + Euler’s Method: from this run the prey and predators are rising and declining in population much more slowly over time since the predators are not hunting their prey as often as previously.
  + 
* The prey breed time is doubled
  + CAM:In this run since the prey take a longer time to breed, when it comes time for the prey to re-populate after being eaten by a large amount of wolves, they take a long time to grow in numbers resulting in the wolves not having enough time to find prey and eat. In result the predator go extinct.
  + 
  + Euler’s Method: with the prey population reproducing so quickly the predators are able to eat as often as they wish resulting in a more rapid change in population over time.
  + 
* The initial amount of animals placed on the same size grid is doubled
  + CAM:In this run the initial amount of animals are increased but the island size is left the same, resulting in the animals eventual extinction since the prey had nowhere to hide from the predators. It was this run that helped lead me to my current conclusion below.
  + 
  + Euler’s Method: With the initial amount of animals doubling, the predator population heads towards 0 as a minimum, meaning predator death.
  + 

**Conclusion:**

From my research, I have found that there are distinct differences between the Cellular Automation Model and Euler’s method, but also many similarities in output. The Cellular Automation Model played out a simulation in real time, hence making it a very delicate system, if either the predator or prey are set to paramaters that are not optimal, then one of them will eventually die out. This is because unlike most mathematical models, this model uses random motion for the animals movements, hence creating a unpredictable outcome that is not controlled. A model that does not return a definite outcome is much more unstable than other models such as Euler’s method. With the Euler’s method runs, the outputs were much more stable, taking into account the previous conditions given by the lotka voltera equation. The only problem I found with Euler’s method however is that when one of the animals reached a population of zero, it would show the population then growing again overtime, obviously this is not the case in a real life scenario. Hence, using the Cellular Automation model may produce a more accurate output, but is much more unstable than the Euler’s methods output, which gave a more stable output but relatively less accurate.