# Excercise 1.
# Implementing a first Application in RePast: A Rabbits Grass Simulation.

Group №10: Cosmin-Ionut Rusu, Sorin-Sebastian Mircea

September 29, 2019

## 1 Implementation

We followed the provided tutorial which served as a base for our implementation.
Grass is coded with green, and the rabbits are coded with blue.

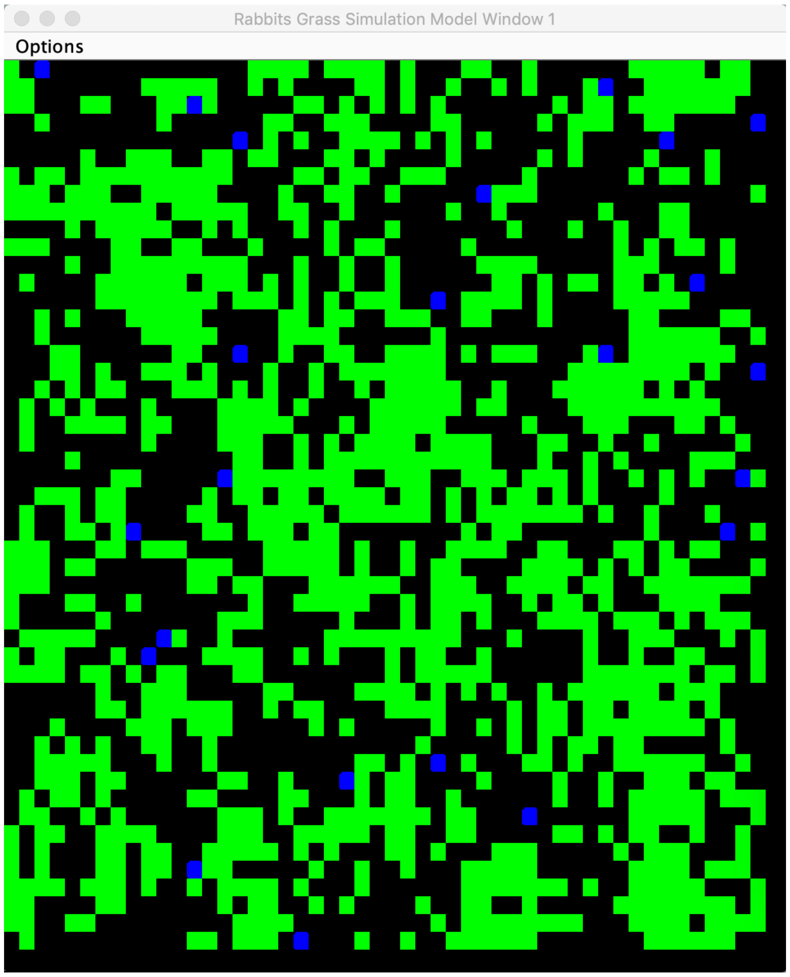


Figure 1: Simulation world

## Charts

In order to better visualise the world we have created two charts:

1. Showing the raport between the tiles of grass available in the world and the population of rabbits. The rabbit population is multiplied by a constant (in our case 100) in order to better depict the variations in population.

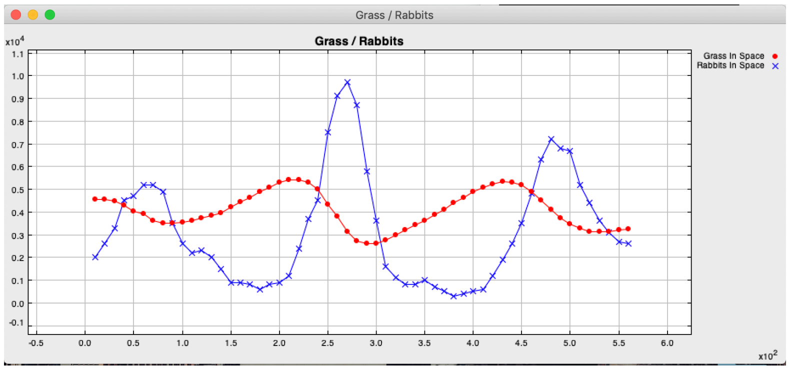


Figure 2: Grass / Rabbits rapport

2. Histogram of the rabbits energy at the current step

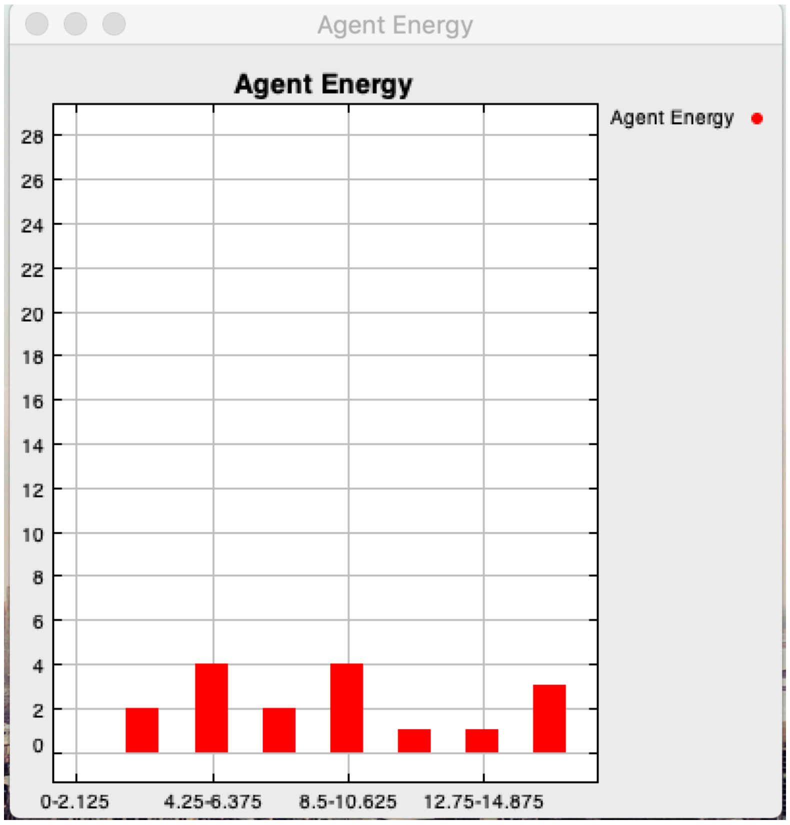


Figure 3: Rabbits energy histogram

### Extra parameters

We considered that the the the following parameters would need to be added (besides the already provided ones) in order to have a better control of the world:

- **EnergyLostByReproduction** - the energy that a rabbit is going to loose in order to reproduce (once it reaches the reproduction threshold)
- **MinRabbitInitEnergy** and **MaxRabbitInitEnergy** - a range depicting the initial energy value of a rabbit; if the range is bigger than one, rabbits are randomly going to receive a value from the interval

## 1.1 Assumptions

### Action order

The order in which our world is going to execute the possible actions can have an impact on the outcome (even this is negligible). Arbitrarily we chose the following order (that is executing during one step):

1. Move agents
2. Agents eat (if they are on a grass)
3. Agents loose energy
4. Agents give birth (if energy ¿= birthThreshold)
5. Agents die (if energy = 0)
6. New grass tiles are spawned

### Grass spawning

We had to decide if we were going to spawn the grass only on the tiles that are unoccupied by the rabbits, or randomly everywhere. We chose the second one, when this happens the rabbits are instantly going to benefit by the extra food. The reason behind this was mainly due to the implementation reasons.

### Initial energy levels

This is a fair assumption since if a rabbit were to be born with an energy level ¿= birthThreshold, then it might reproduce right away and we will have more rabbits on the space.

### Rabbit in impossibility of movement

In the case that a rabbit cannot move due to it being surrounded by other rabbits, we chose for the rabbit to stay on place but still loose energy (as a penalty for not being able to search for food).

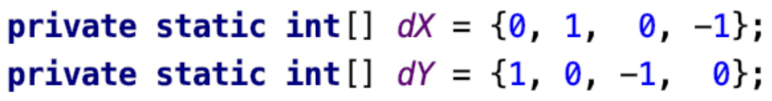
### Collision system

Rabbits cannot collide with other rabbits (during spawning and movement), it can move onto a grass tile (though, at that moment the grass will be consumed).

## 1.2 Implementation Remarks

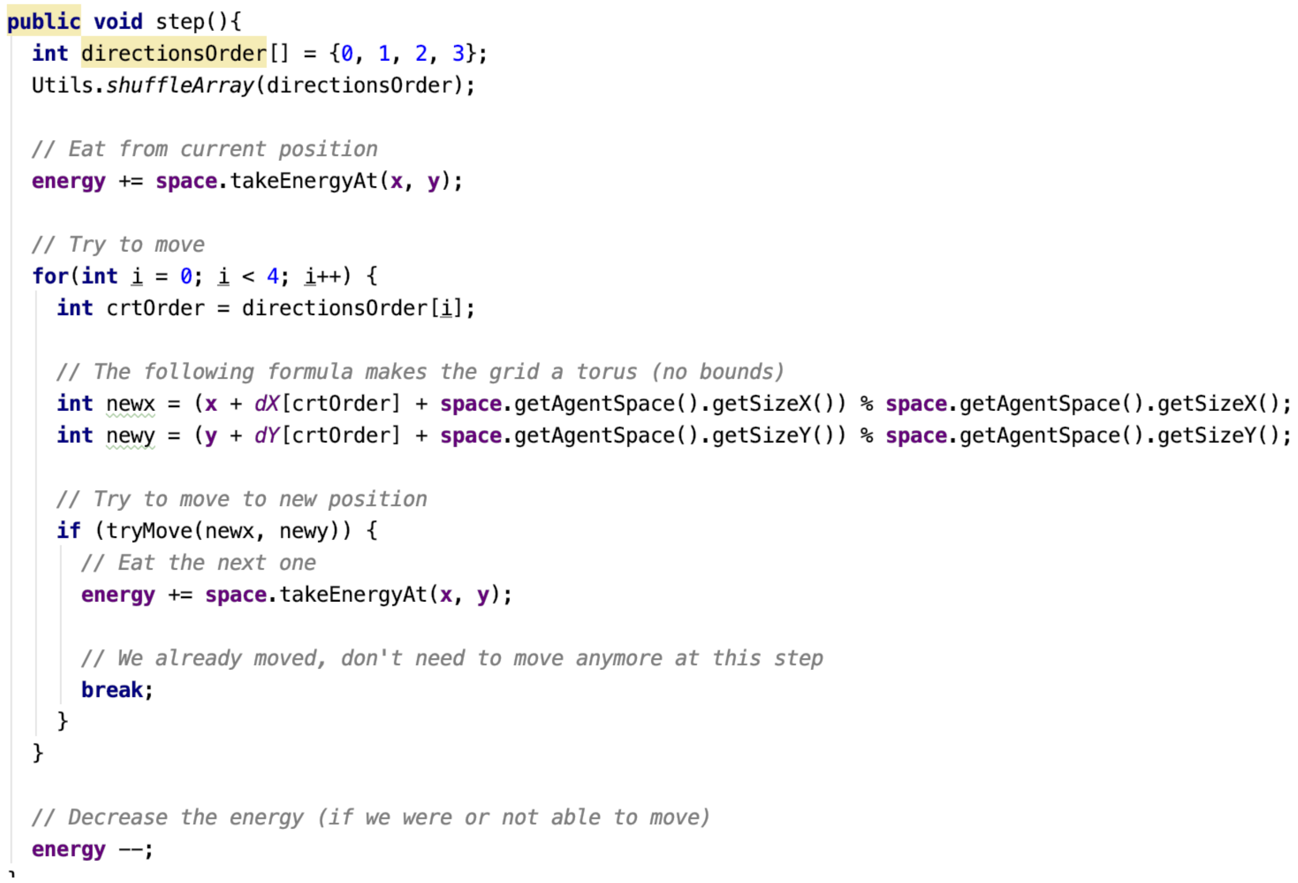
Grass tiles cannot collide with other grass tiles (during spawning). Grass can be spawned on top of a rabbit but it will be consumed instantly by the rabbit.

## Movements



```java
private static int[] dX = {0, 1,  0, -1};
private static int[] dY = {1, 0, -1,  0};
```

Figure 4: Directions of movement (Right, Down, Left, Up)



```java
public void step(){
  int directionsOrder[] = {0, 1, 2, 3};
  Utils.shuffleArray(directionsOrder);

  // Eat from current position
  energy += space.takeEnergyAt(x, y);

  // Try to move
  for(int i = 0; i < 4; i++) {
    int crtOrder = directionsOrder[i];

    // The following formula makes the grid a torus (no bounds)
    int newx = (x + dX[crtOrder] + space.getAgentSpace().getSizeX()) % space.getAgentSpace().getSizeX();
    int newy = (y + dY[crtOrder] + space.getAgentSpace().getSizeY()) % space.getAgentSpace().getSizeY();

    // Try to move to new position
    if (tryMove(newx, newy)) {
      // Eat the next one
      energy += space.takeEnergyAt(x, y);

      // We already moved, don't need to move anymore at this step
      break;
    }
  }

  // Decrease the energy (if we were or not able to move)
  energy --;
```

Figure 5: Rabbit procedure of doing one step

## 2 Results

### 2.1 Experiment 1 variating amount of initial rabbit ( 500 steps)

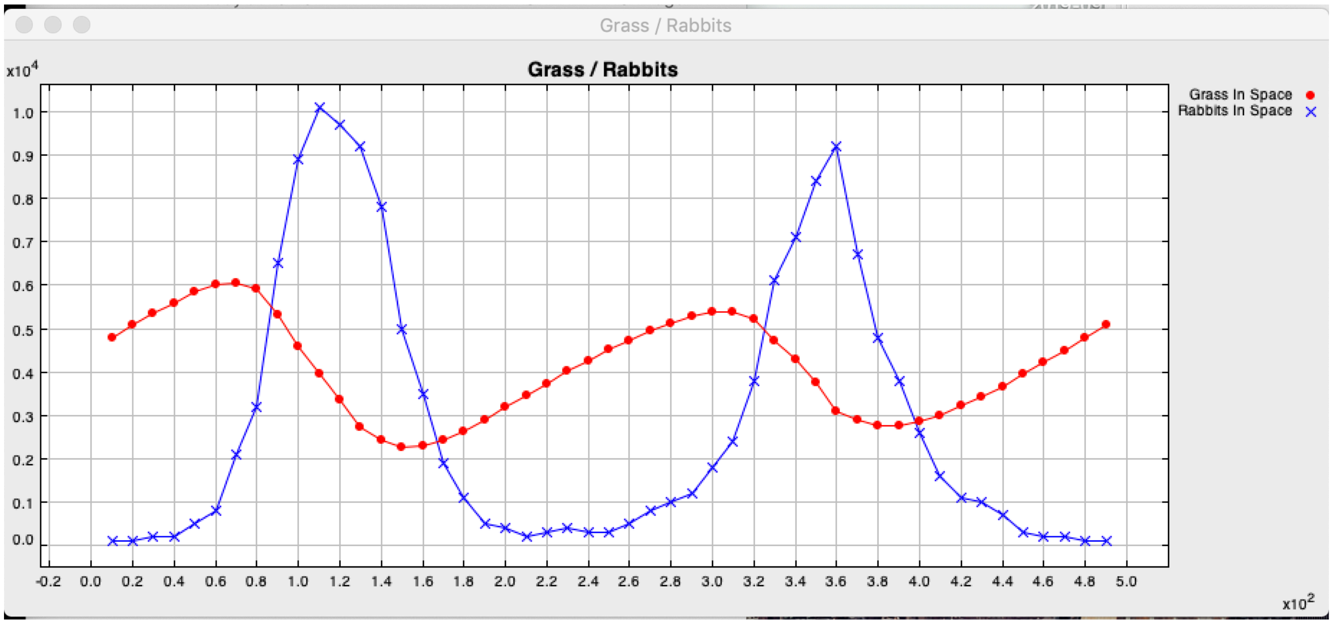


Figure 6: 1 initial rabbit

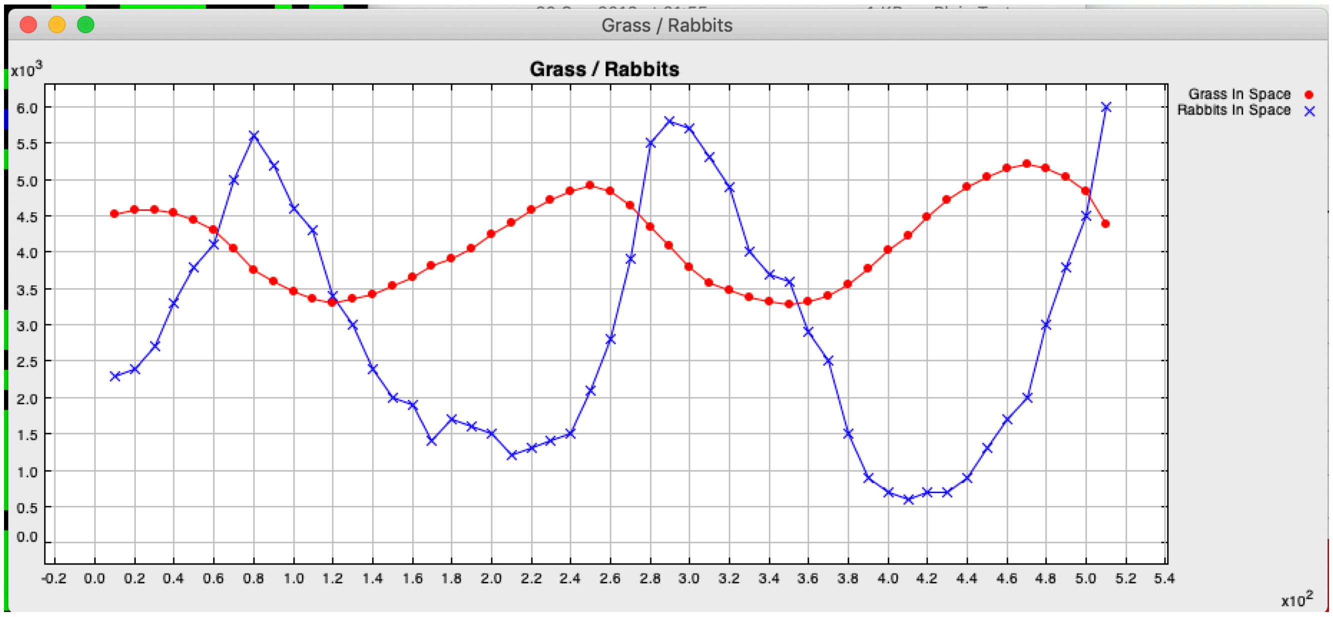


Figure 7: 20 initial rabbits

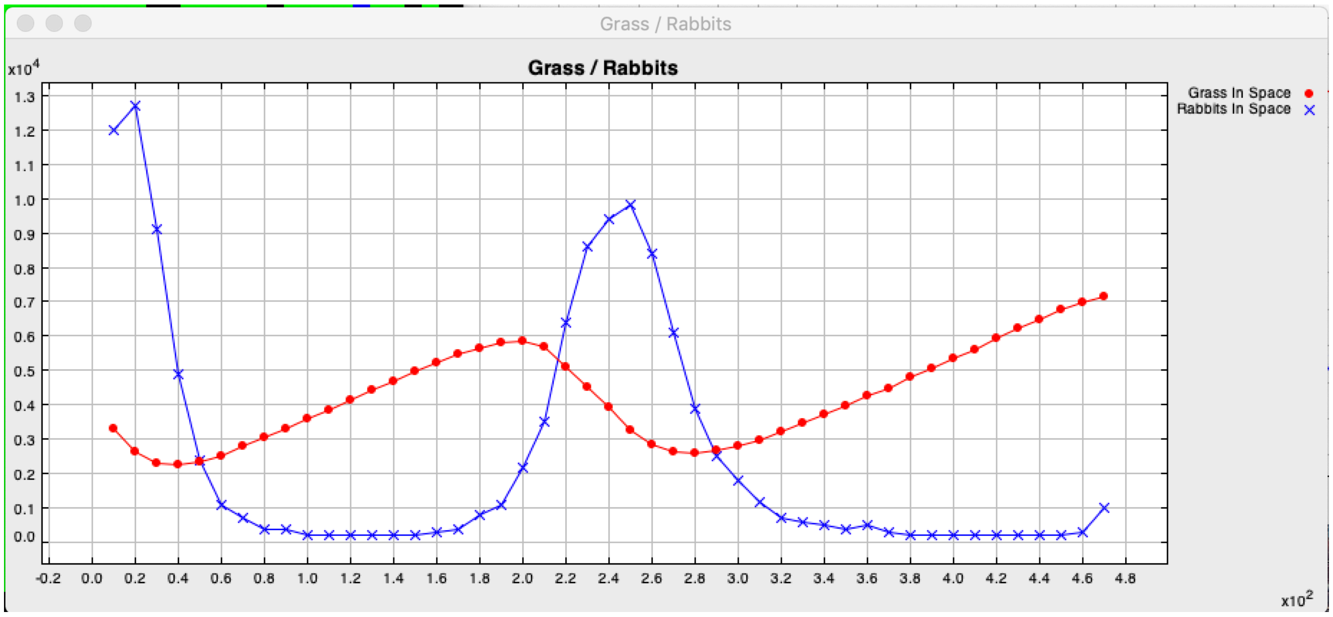
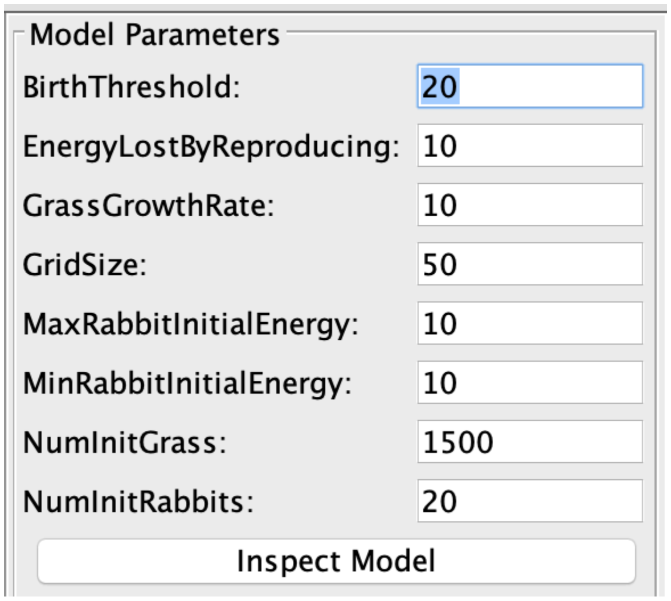


Figure 8: 100 initial rabbits

### 2.1.1 Setting



Model Parameters

| | |
|---|---|
| BirthThreshold: | 20 |
| EnergyLostByReproducing: | 10 |
| GrassGrowthRate: | 10 |
| GridSize: | 50 |
| MaxRabbitInitialEnergy: | 10 |
| MinRabbitInitialEnergy: | 10 |
| NumInitGrass: | 1500 |
| NumInitRabbits: | 20 |

Figure 9: Ex 1 params

### 2.1.2 Observations

We can observe that variating the initial amount of rabbits does not impact the population on long term; there are still going to be periods of great decline in population (when food is going to drastically increase) followed by a population increase and the decrease of food resources.

## 2.2 Experiment 2 variating the energy lost through reproduction

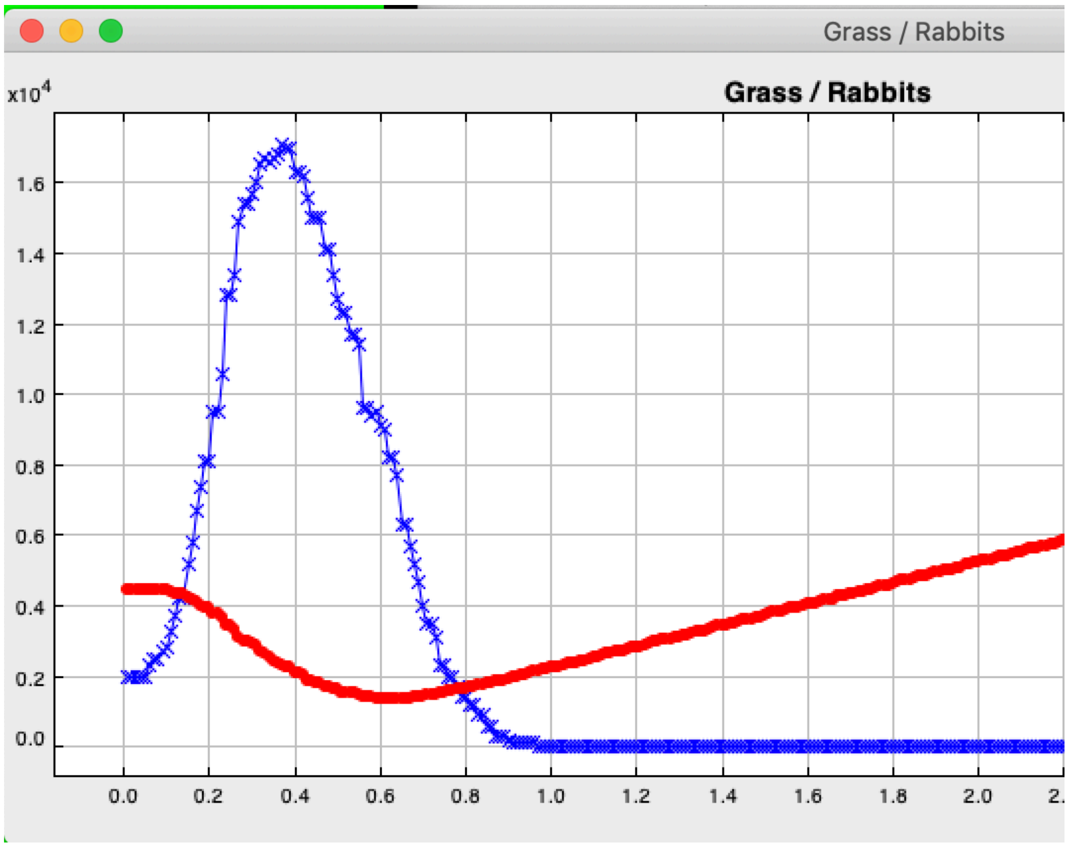


Figure 10: cost of reproduction is 1

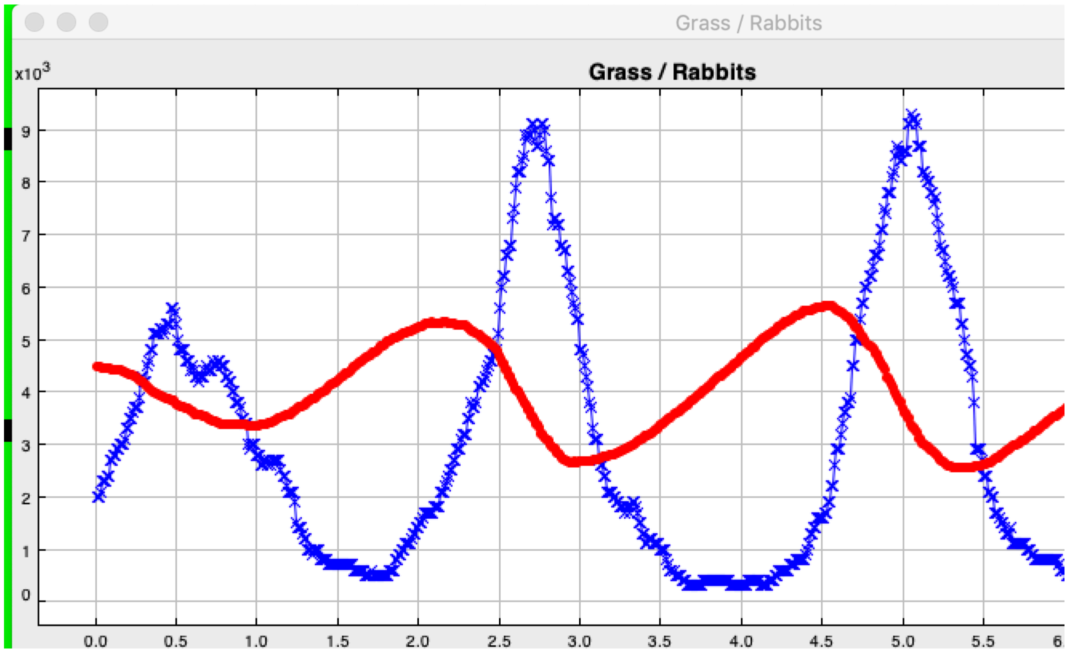


Figure 11: cost of reproduction is 10

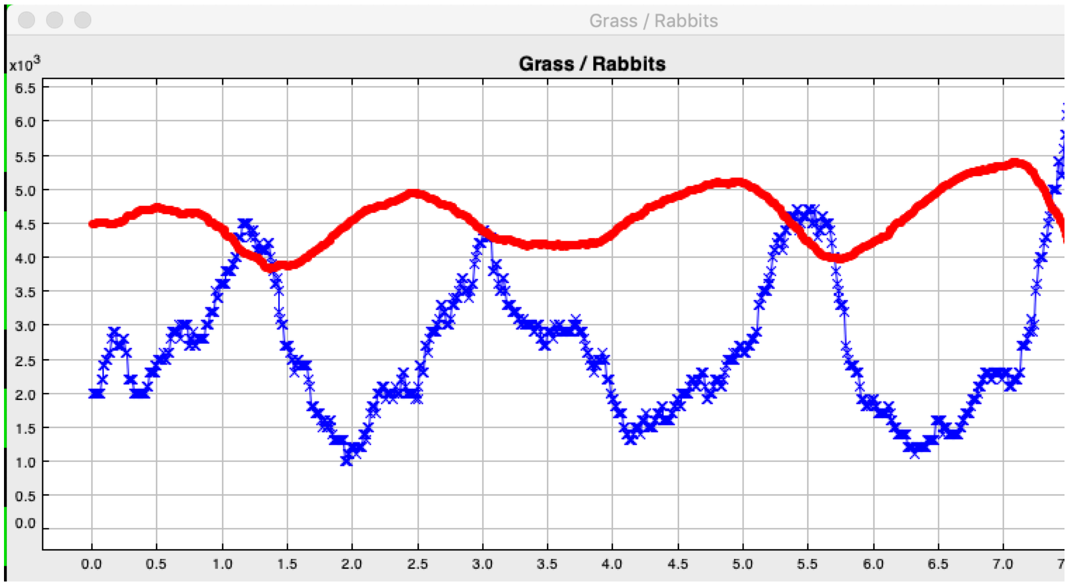


Figure 12: cost of reproduction is 15

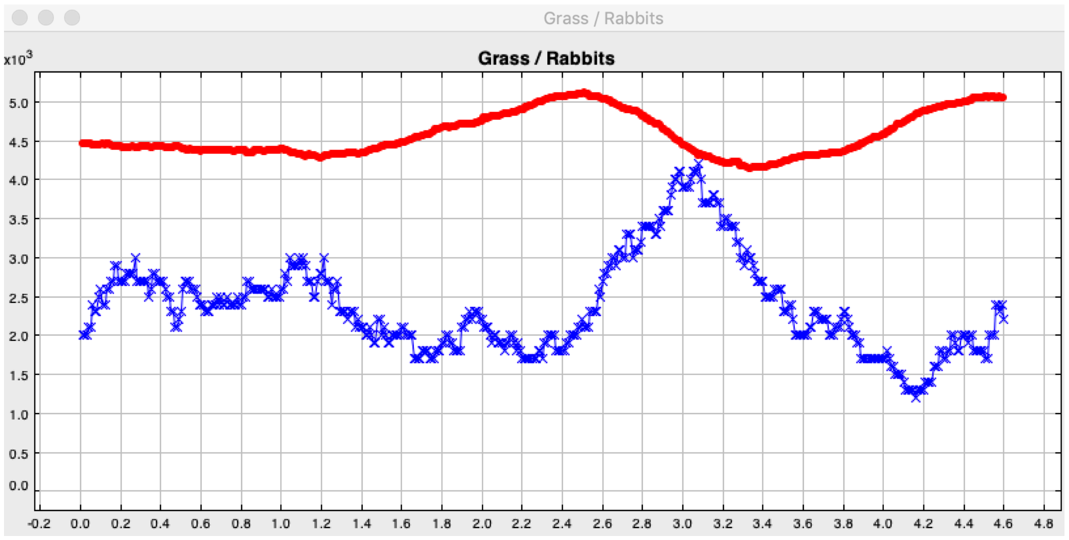
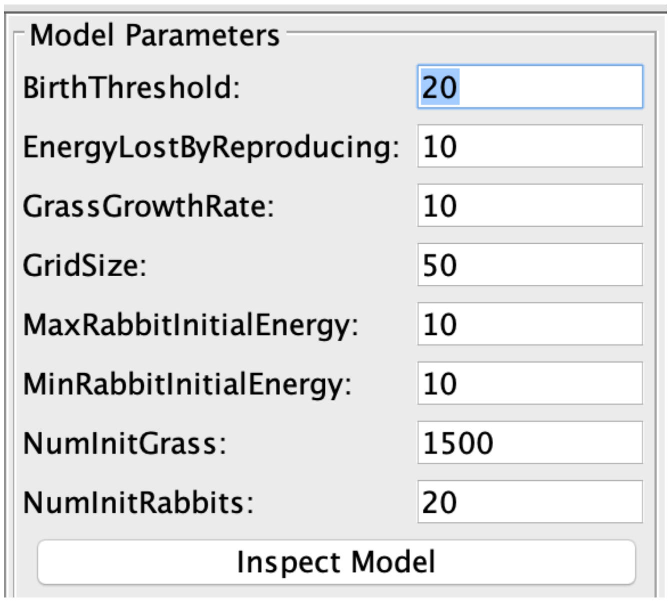


Figure 13: cost of reproduction is 18

### 2.2.1 Setting



Model Parameters

| | |
|---|---|
| BirthThreshold: | 20 |
| EnergyLostByReproducing: | 10 |
| GrassGrowthRate: | 10 |
| GridSize: | 50 |
| MaxRabbitInitialEnergy: | 10 |
| MinRabbitInitialEnergy: | 10 |
| NumInitGrass: | 1500 |
| NumInitRabbits: | 20 |

Figure 14: Ex 1 params

### 2.2.2 Observations

As the cost of reproduction increase the amount of available grass increases too.

The first simulation (the one with cost of reproduction 1) has an interesting results, as the rabbit population booms too fast it exhausts the available grass (not allowing enough time for more grass to grow) and this leads to extermination for rabbits.

## 2.3 Experiment n

### 2.3.1 Setting

### 2.3.2 Observations