Implementing a first Application in RePast: A Rabbits Grass Simulation

Ana Manasovska, Valentin Rutz

September 2015

1 Description of the code

1.1 RabbitsGrassSimulationAgent

This class represents a rabbit. It contains its position and the current amount of energy in that agent. The actions that it can perform are gain/lose energy and move to a neighboring position in the agent space.

1.2 RabbitsGrassSimulationSpace

This class is used to manage the states of the both spaces, Agent Space and Grass Space. It stores and edits the values of every position in the grid. The Grass Space stores how many grass in on every position in the grid and the Agent Space stores the position of the agents in the grid.

1.3 RabbitsGrassSimulationModel

This is the main class of the program. The program execution is divided in three parts:

Build Model: This function is executed on the start of the program and we use it create the both spaces and we initialise the grid.

Build Display: We map the values to different colors and we make everything displayable.

Build Schedule: The schedule is the most important function in order for the simulation to work. Here is the code that is executed on every step. First, if we don't have any rabbits left, we stop the simulation.

For each rabbit, we ask the rabbit to try to move to a random neighboring position and if there is a free position and if manage to find it in finite number of trials, it is going to move to it. If there is grass on that position the rabbit eats it.

After the that, we check the rabbit's energy level. If it is below 0, it means that the rabbit is dead and we remove it from the Agent Space and the list.

If the rabbit is alive, we check if its energy level is above the birth threshold, that would mean that it can reproduce. Reproduction means we reduce its energy by reproduction cost.

When rabbit reproduces we place a new rabbit in the agent space, on a random position that is not occupied by another rabbit. It there is grass on that position, the new rabbit eats it. The newborn rabbits don't move in the same step when they are born.

For language specific reasons, we had to use an Iterator<RabbitsGrassSimulationAgent> to iterate through the list of all agents in order to remove the dead ones while iterating. But

we are not able to add new agents directly in the list we are iterating, so we add them in a temporary list and after the end of the iteration we copy the list to the original.

Rabbits can lose energy in several ways: moving, reproducing and living. Moving and reproducing are logical costs but the cost of living was added to deal with the situation where there were only rabbits on the grid and no more grass nor free space. The rabbits would never move and thus never lose energy and that meant that the rabbits would live forever because they were not moving which seems illogical. This is why the cost of living was introduced.

2 Results

We have noticed that is not straight forward to have stable system with balanced number of rabbits and grass. It is usually the case that the one population takes over the other, as it show on the graph below.

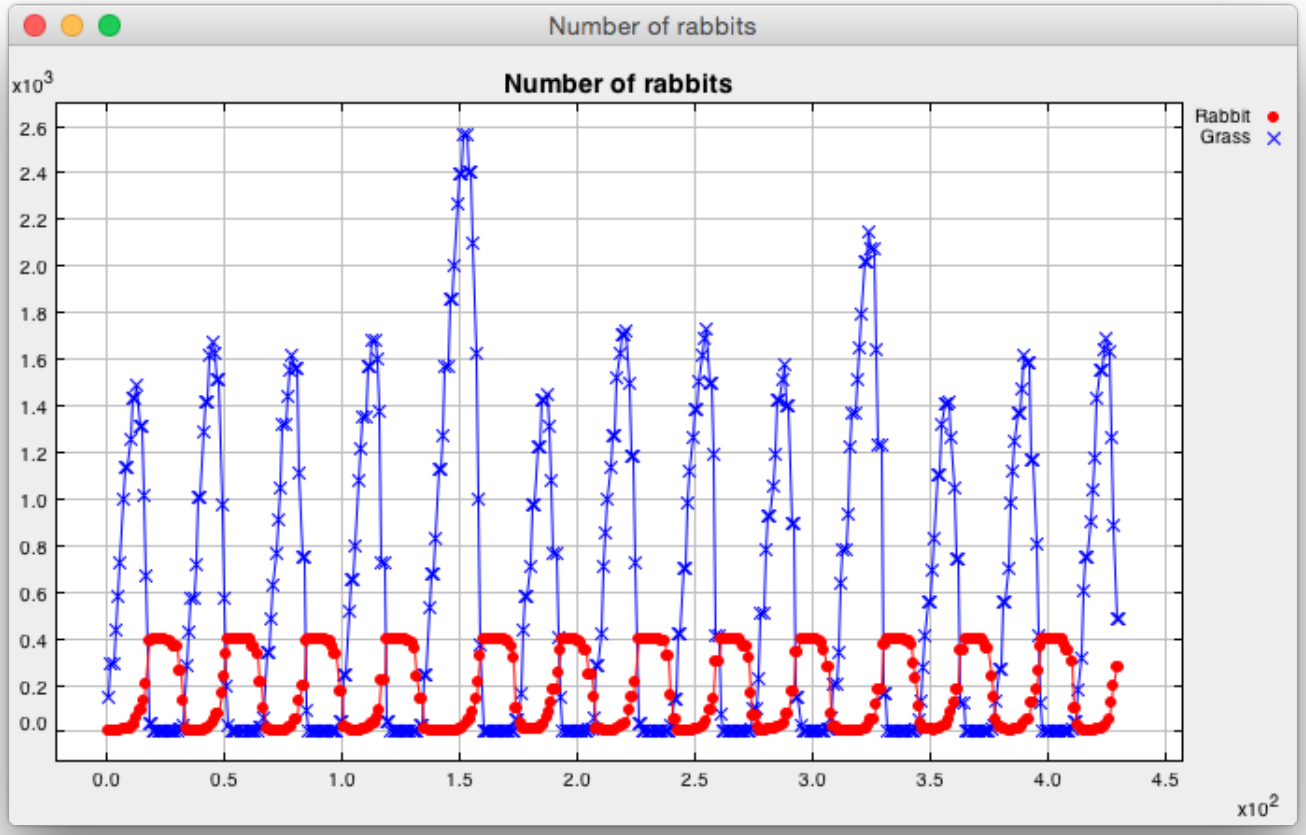

Figure 1: Number of rabbits and amount of grass over time

Parameters that have impact on the graph:

Grass growth rate If the rate is very high, the rabbits population and the grass population are almost stable (see fig 2)

Reproduction cost Depends on the birth threshold. If too high compared to the birth threshold, the rabbits would give birth to a newborn and almost immediately die and it never overpopulates (see fig 3).

Initial energy Depends on the birth threshold. If it is too low compared to the birth threshold, the rabbits might never reproduce and the population would decrease quickly to extinction.

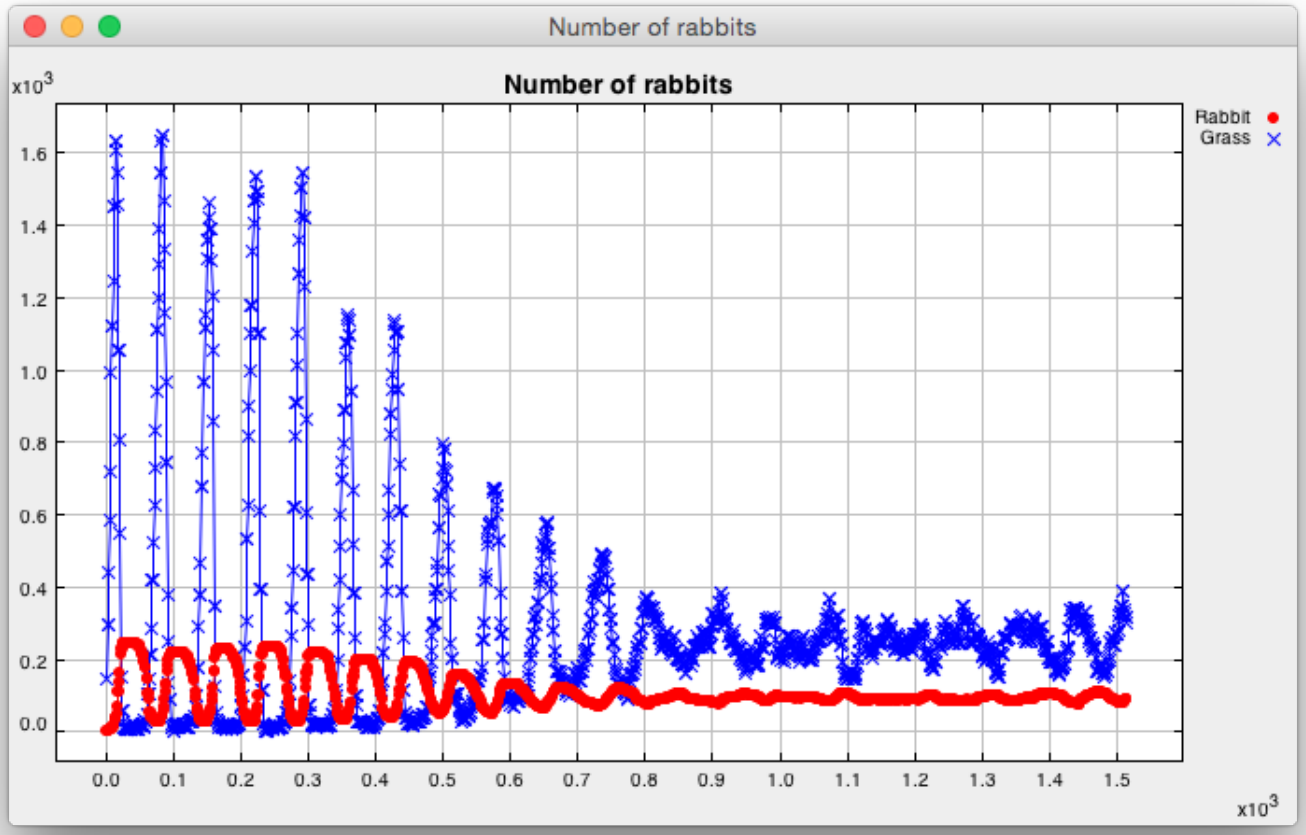

Figure 2: Number of rabbits and amount of grass over time

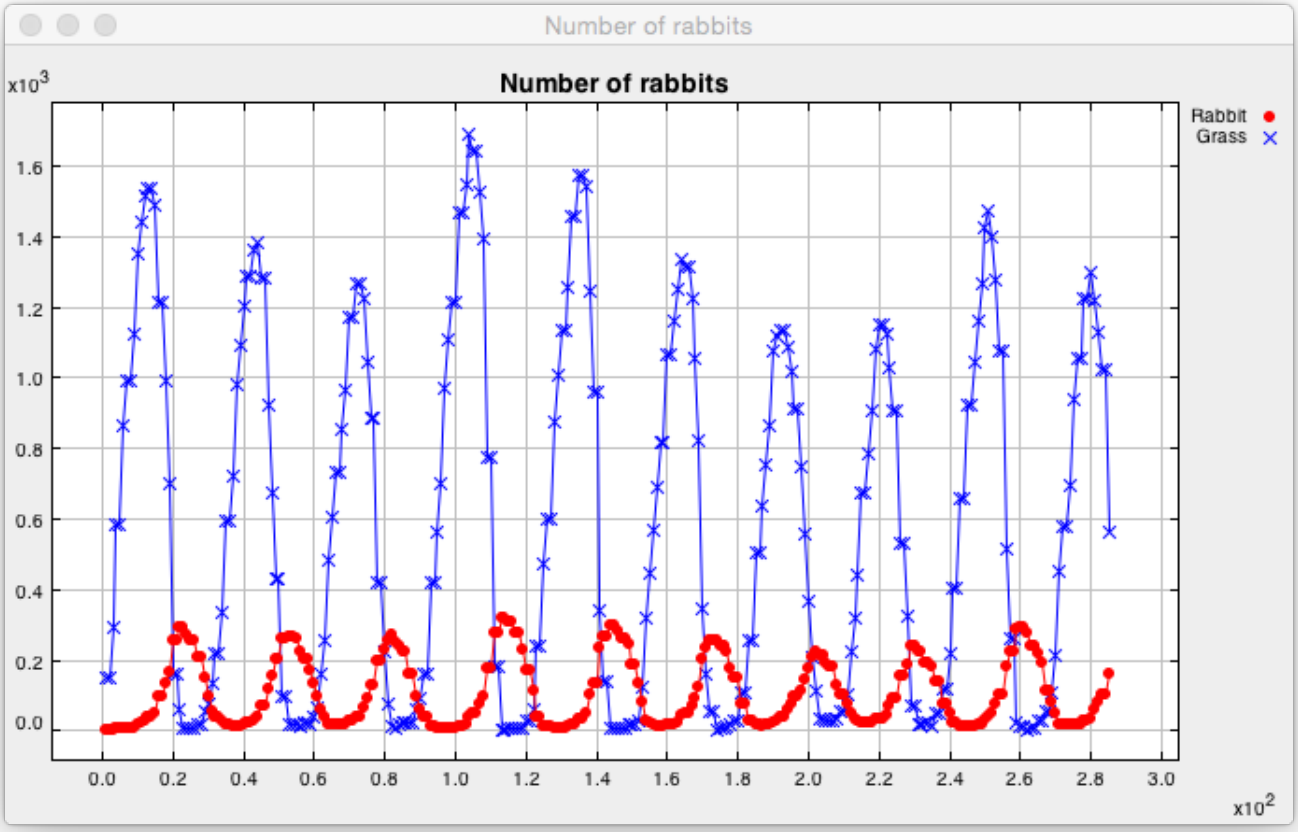

Figure 3: Number of rabbits and amount of grass over time