# EEN118 LAB EIGHT

One of the most important uses of computers is in simulation. Some experiments are too dangerous, too slow, too expensive, or too cruel to perform on real subjects, but if you know (or just think you know) how a particular real-life system works, it is possible to construct a model. A model is a program that behaves in the same way as the real-life system, but perhaps many times faster and without the risks associated with really doing something. If your model is correct, it will tell you how the real system would behave under the same conditions. If you're not sure about your model, comparing its behaviour with real life will tell you whether its right or not.

We will simulate the ecology of predator and prey species interacting in some natural environment. This is too slow for real experiments, as many years are required to see long-term effects, and interfering with the food supply to see when each species would starve is generally considered quite unethical.

We will simulate an ecological system in three steps. First some rabbits in an environments with an unlimited food supply, where the only factor affecting their success as a species is their ability to reproduce faster than they die of natural causes. Then we will place limits on the food supply and see what difference it makes. Finally we will introduce some foxes to eat the rabbits and see how the two species coexist.

Our simple model for rabbits living with an unlimited food supply is that each year every rabbit has a certain number of offspring (given by the birth rate), and a certain fraction of the rabbits die (given by their birth rate). The user who is performing experiments with the system will provide the initial conditions (how many rabbits there are at time zero), and the parameters of the simulation (the birth and death rates) and see what happens.

The simulation will work on a year-by-year basis, which is pretty much the way most wild animal populations are, calculating the new rabbit population after each step and displaying the ongoing results.

The number of rabbits is obviously an integer, but the birth and death rates must be floating point numbers. The birth rate can be any positive number, 0 would produce a totally sterile population, and 2.4 (for example) would mean that the average bunny has 2.4 surviving babies each breeding season. The death rate can only vary between 0 (for universal immortality) and 1 (meaning no rabbit survives beyond their first year). In real life there would be a multitude of additional parameters describing weather, evil spirits, diseases, and who knows what else.

The state of the world should be represented by a few variables:

```
int num_rabbits;
double rabbit_birth_rate, rabbit_death_rate;
```

Each year in our utopian rabbit world is modelled by a very simple function. Multiplying the current population by `(1.0+rabbit_birth_rate)` takes account of new births, and multiplying by `(1.0-rabbit_death_rate)` takes account of the deaths, so a simple little function acting on the global variables models the passing of one year:

```
num_rabbits *= 1.0 + rabbit_birth_rate;
num_rabbits *= 1.0 - rabbit_death_rate;
```

1. ***(Basic Required)*** *Complete the Program*

    Build this simple plan into a whole program that first asks the user to input the three essential numbers, then simulates the passage of 20 years (or more if you prefer). You should print out the current population after each simulated year so that you can see just how the numbers vary.

    If the population ever drops to zero, you know it isn't going to recover, so the simulation should stop at that point. We also have to guard against incorrect results caused by overflow. If the population ever exceeds 100,000,000 or so, it is probably a good idea to stop too.

    Even with this simplistic model you should be bale to see three distinct behaviours. If the birth rate is at all healthy, the population will explode very rapidly. If the death rate is too high, the rabbits will become extinct very quickly. If the birth and death rates are in perfect balance the population remains constant. Make sure you can produce each of these three behaviours with suitable non-extreme choices for the birth and death rates.

2. ***(Basic Required)*** *Better Visualisation*

    Most people don't like staring at rows of numbers; everyone likes nice clear graphics. A picture is worth a thousand words. Make your program draw a graph of the world's rabbit population as the simulation progresses.

    The horizontal axis should be the number of years that have passed, and the vertical should be the population of rabbits. You will certainly need to stretch the horizontal scale: you'll need *at least* ten pixels per year to be able to see the graph properly. You will probably want to scale the vertical axis too. You've already seen how rapidly the population can grow.

3. ***(Advanced)*** *Clever Scaling*

    How can you possibly come up with a good vertical scale? Some parameters result in the population rocketting into the millions in a few years, some reduce it to zero just as quickly. How could you ever see both behaviours clearly with the same kind of graph?

    There are two solutions. One you've already seen in a previous lab: run the simulation once without drawing anything, just to see what the minimum and maximum values are, then work out the scale factors and run the simulation again to draw the graph.

    Another option is to use a logarithmic scale for the population axis. When a number `x` varies between 1 and 10, its common (base 10) logarithm varies between 0 and 1. When the number varies between 1,000,000 and 10,000,000, its common logarithm varies between 6 and 7. The whole range of `int`s is reduced to the range 0 to 10 by taking the base-10 logarithm. Remember that the base-10 logarithm function in C++ is called `log10`. So `50*log10(x)` will nicely fill a window 500 pixels high, with nice visibility at every scale. Try it.

    If you use a logarithmic scale for population, all the graphs will come out as straight lines. Think about what that means.

4. ***(Basic Required)*** *Hunger*

    The only thing that a rabbit will eat is grass. Every rabbit requires two units of pasture land to maintain average health and happiness. If the rabbits have less food than they need, their health will suffer, and the birth rate will decrease and the death rate will increase. If they have more food than they need, they'll produce more offspring. If you are an expert biologist you will probably be able to come up with a formula that you think describes this relationship. If you aren't, we'll just have to make somethng up. Surprisingly, any half-way reasonable formula gives very realistic results. We'll define a *Hunger Factor*, which should obviously be computed afresh for each simulated year:

```
rabbit_hunger = num_rabbits/pasture_area + 0.5;
```

The logic of this formula may escape you, but consider a few simple examples. Remember every rabbit wants 2 units of pasture area. If they have that (e.g. `num_rabbits=100, pasture_area=200`) then the formula gives a hunger factor of 1.0, a nice central value. If they have more food than they need, the hunger factor is less than 1.0, if they have less, it is more than 1.0. Having a measurement whose normal value is 1.0 makes things very easy.

How is hunger taken into account by the model? Very simply, when each year is simulated, if the hunger factor is greater than 1.0, the effective death rate is multiplied by it. If the hunger factor is less than 1.0, the effective birth rate is divided by it. If you've got a better idea of how it might work, you can try your idea instead.

Add `pasture_area` as a new variable that the user supplies a value for before the simluation begins, and explore the differences it makes to our model.

You may be surprised by the difference this one new factor makes. You shoild now be able to find five distinct behaviours for the system:

1. Too low birth rate produces a rapidly declining population until extinction.
2. Moderate birth rate produces smooth growth to a stable maximum population.
3. Quite high birth rate produces rapid growth followed by wild oscillations before stability is reached.
4. Too high brith rate produces seemingly random populations each year: wild oscillations that never reach a stable healthy population.
5. Some high values for brith rate produce wildly oscillating populations that accidentally make themselves extinct at some point.

Even the existence of these five modes of behaviour is not obvious without the aid of computer simulations. Try to find values of the parameters that will let you see each of the five possibilities (number 5 requires a little luck). All five of these modes of behaviour are found in nature exactly as our absurdly simplistic model predicts.

High birth rates (modes 3, 4, and 5 in the list above) produce what are called *Chaotic Systems* (just like in Jurassic Park, but this time for real). Animals in chaotic populations are generally believed to be highly stressed, unhealthy, and as far as animals can be, unhappy.

## *5. **(Normal)** Predators*

Foxes actually. Introduce a second species into the simulation with another variable, `num_foxes`. The initial value for the fox population should be supplied by the user before the simulation starts, and of course should be a lot less than the rabbit population.

The fox population behaves exactly the same as the rabbit population. They have their own birth and death rates (two more global variables) and their own hunger factor. Except of course the hunger factor for foxes depends upon the number of rabbits per fox, and has nothing to do with the amout of grass. How many rabbits does each fox need to keep it in perfectly balanced health? How should I know? We'll have to make it another parameter supplied by the user so we can experiment with it and see what happens.

If you have a variable called `rabbits_eaten_per_fox,` the hunger factor for foxes would be calculated by a formula something like this:

```
fox_hunger = num_foxes * rabbits_eaten_per_fox / num_rabbits;
```

this formula again gives a value of 1.0 for "just right", more than 1.0 for "hungry", and less than 1.0 for "over fed".

In ever year of simulation both the rabbit hunger factor and the fox hunger factor should be computed, and the rabbit and fox populations should both be adjusted accordingly. Remember to subtract the number of rabbits that get eaten from their population.

Plot both populations, using a different colour for foxes and rabbits so you can see which is which.

Run the simulations again, and see what different behaviours you can produce.

You should observe two interesting things, that are both seen in nature. When things are nice and stable, the rabbit population peaks after a few years before reaching a steady state. The fox population peaks a couple of years later, then also reaches a steady state.

You should also see that it is much harder to come up with parameters that induce chaotic behaviour in the rabbit population. Think about what that means: the rabbits will be healthier and happier when there are foxes around trying to eat them.

*6.* ***(Advanced)*** *Monsters****.***

Add a third species, a super-predator, that lives by eating foxes. Perhaps they are tigers or something, or even worse: people! See what their presence does to the whole system. You may be surprised.

Think about what you have seen. You may think the formulas I suggested were too simplistic, or just plain wrong, but the fact is that any reasonable formulas for hunger, birth, and death will result in more or less the same results. Predators are good for the prey. In a popluation without predators growth is limited by the food supply. If the birth rate is too high, the population becomes chaotic, alternating between famine and near-extinction. If the birth rate is too low, extinction is inevitable. If the birth rate is within an easily determined healthy range, its growth is clearly controlled by the food supply.

What does this tell us about the human condition? What are our chances of finding a solution to world hunger? Would we indeed be better off as a species if some lunatic did clone some angry dinosaurs to chase us around a bit?

How could we ever learn about these things without computer simulations? It would take literally centuries horrendously expensive of cruel experiments with large populations of real animals. With a computer, we can first compare our model's predictions to real life, to make sure we've got it right, then use the model to see what would happen in situations we can't reproduce.

This is how weather forecasting, stock predictions, and all sorts of other things, both valid and foolish, are done.