Francesca Cuteri: cuteri@th.physik.uni-frankfurt.de

Final Project: Predators *vs.* preys

"Your almost unique opportunity of writing a code in which you are not allowed, but rather supposed to have "bugs"... (🐞 🐝 🐞 🐝) ...which you should not try to get rid of by assuming constant zero populations!"

1 Introduction

In this project we are interested in a model, the *Lotka-Volterra* model, describing the population growth, in a closed ecosystem, of two species of animals in which one of the two species (the *preys*) is the primary food source for the other species (the *predators*).

There are many examples of situations of which the model we aim to implement can provide a semi-realistic description: baleen whales *vs* Antarctic krill (in turn eating plankton) in the Southern Ocean, wolves *vs* rabbits (in turn eating vegetation) in a closed forest.

Figure 1: **Cottony cushion insect (Icerya Purchasi)**

Remarkably the model we are going to deal with was even able to explain observations in the behavior of the involved species when, in 1968 in California the cottony cushion insect (Icerya Purchasi) was accidentally introduced from Australia and it was threatening all citrus crops. To solve the problem it was imported in 1888-1889 by C. V. Riley, again from Australia, the ladybird beetle (Novius Cardinalis), the most effective predator for the cottony cushion insect. It was one of the first major successes of biological control resulting in swift reductions of the cottony cushion insect populations, saving the burgeoning Californian citrus industry from this destructive pest.

Figure 2: **Ladybird beetle (Novius Cardinalis)**

However, later on, when the insecticidal action of the Dichlorodiphenyltrichloroethane (DDT) was discovered in 1939, farmers started using it in the hope of reducing even further the cottony cushion insect population. However, DDT turned out to be fatal to the beetle as well, and the overall effect of using the insecticide was to increase the numbers of the cottony cushion insect. Also this aspect of the predator-prey dynamics and the related experimental observations can be explained by slightly modifying the Lotka-Volterra model to keep the DDT effect into account. Yet another big success for the model.

Still biologist and ecologist have reason to criticize the Lotka-Volterra model for being unrealistic because the system, as described by the model, is not asymptotically stable, whereas most natural predator-prey systems tend to reach equilibrium levels over time after showing regular population cycles. This is why alternative models that are asymptotically stable have been proposed where also the degree of competition among the predators for the finite amount of available preys as well as the degree of internal competition among preys due to the limited amount of resources (e.g. food) are taken into account.

2 Overview of the problem

Let us take a closer look at the ecosystem of the ladybird beetles and cottony cushion insects. The beetles eat the insects, and the insects live on the citrus crop. If the beetles eat so many insects that these latter

cease to be abundant, the food supply of the beetles is greatly reduced. Then the ladybird beetles will starve. As the population of beetles dwindles, the cottony cushion insects population makes a comeback because not so many of them are being eaten any more. As the insects population increases, the food supply for the beetles grows again and, consequently, so does their population. Also, more beetles are eating increasingly more insects again.

In the closed ecosystem, does this cycle continue indefinitely or does one of the species eventually die out? What effect does exploitation of pesticides have on the balance between the ladybird beetle and cottony cushion insect populations? The ability to answer such questions is important to management of biological control as alternative to other methods for the protection of farming plantations. Answers can be obtained from a computational approach to solve numerically the system of differential equations describing the model.

3 Overview of the model

Let $x(t)$ and $y(t)$ indicate, at any time t, the cottony cushion insect and the ladybird beetle population respectively within a closed Californian citrus crop.

The level of the insect population $x(t)$ depends on a number of factors, including the ability of the crop to support them, the existence of competitors for their primary food source (the citrus plants) and the populations of other possible predators than the beetle. We will however start building the model under the assumption that our unfortunate citrus crop can support an unlimited number of insects so that their population could in principle grow exponentially if it was not for the beetles that we assume to be the unique predators of the insects. This reduces the growth rate of the insects in a way that is proportional to the number of interactions between them and the beetles. All above observations leads to the differential equation

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = [a - b\,y(t)]\,x(t)\ , \tag{1}$$

where a, b are positive constants (determined on the basis of observations) indicating the degree of self-regulation of the insects population given the unlimited amount of resources and the predatoriness of the beetles, respectively. So $a - b\,y(t)$ is the intrinsic growth rate of our population of preys.

About the level of the ladybird beetle population $y(t)$, one can observe that in the absence of their primary food source $y(t)$ declines at a rate which is proportional to their numbers, i.e. according to exponential decay kind of equation. However, in the presence of preys, $y(t)$ is also expected to increase at a rate which is proportional to the interactions between the two species, boiling down to another differential equation

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = [-m + n\,x(t)]\,y(t)\ , \tag{2}$$

where m, n are positive constants (determined on the basis of observations) and $-m + n\,x(t)$ is the intrinsic growth rate of our population of predators. So for our predator-prey model we obtained an autonomous system of differential equations

$$\begin{cases} \dfrac{\mathrm{d}x(t)}{\mathrm{d}t} = [a - b\,y(t)]\,x(t) \\ \dfrac{\mathrm{d}y(t)}{\mathrm{d}t} = [-m + n\,x(t)]\,y(t)\ , \end{cases} \tag{3}$$

where $x(0) = x_0$, $y(0) = y_0$ and a, b, m, n are all positive constants. These is the so-called Lotka-Volterra system of equations, independently developed in 1926 by the Italian mathematician Vito Volterra (1860-1940) and American statistician Alfred J. Lotka (1880-1949). It describes the interaction of the ladybird beetles and the cottony cushion insects under the limited growth assumptions made above and in the absence of other competitors/predators.

The system of differential equations in Eq.(3), characterizing the basic Lotka-Volterra model, can indeed be analytically solved and two important properties of the model that can be easily proven are that:

(i) The system admits cyclic solutions (predators and preys sustain each other) of period T;

(ii) The averages of the two populations depend on the parameters a, b, m and n

$$\begin{aligned} \langle x \rangle &= \frac{1}{T} \int_0^T x(t) \mathrm{d}t = \frac{m}{n} \\ \langle y \rangle &= \frac{1}{T} \int_0^T y(t) \mathrm{d}t = \frac{a}{b}. \end{aligned} \tag{4}$$

We want to solve the model numerically and check the above properties.

Later on, we also want to explain effects from the employment of DDT in citrus crops in terms of Lotka-Volterra laws. To do so, we have to modify the model to keep into account that the insecticide destroys in the same way both the preys and the predators modifying the equations for the system to

$$\begin{cases} \dfrac{\mathrm{d}x(t)}{\mathrm{d}t} = [a - \epsilon - b\, y(t)]\, x(t) \\ \dfrac{\mathrm{d}y(t)}{\mathrm{d}t} = [-m - \epsilon + n\, x(t)]\, y(t)\ . \end{cases} \tag{5}$$

This means, for the second Lotka-Volterra law, i.e. according to Eq. (4), that the average number of beetles has decreased, becoming $\frac{a-\epsilon}{b}$, whilst the number of preys has increased to $\frac{m+\epsilon}{n}$. This was indeed observed experimentally and was one of the big successes of Lotka-Volterra systems.

As a last step towards a more realistic predator-prey model, the internal competition of the preys for their limited amount of resources as well as the competition among predators due to the finite amount of available preys can be encoded in the original Lotka-Volterra model getting to yet another system of equations

$$\begin{cases} \dfrac{\mathrm{d}x(t)}{\mathrm{d}t} = [a - b\, y(t) - r\, x(t)]\, x(t) \\ \dfrac{\mathrm{d}y(t)}{\mathrm{d}t} = [-m + n\, x(t) - s\, y(t)]\, y(t)\ . \end{cases} \tag{6}$$

Solving numerically this last model you will be able to check that the trajectories that constitute a solution to this model are not periodic and tend to equilibrium levels.

4 Integrating the relevant Ordinary Differential Equations

All the dynamics in our model is encoded in the system of first-order ordinary differential equations in Eq. (3). Both equations relate a total derivative to some function of the dependent variables $x(t), y(t)$ where the independent variable does not appear explicitly. What we will solve numerically is the corresponding first-order initial value problem (IVP) given the equations, to be solved simultaneously, and as many initial boundary conditions. Not requiring an exact solution function to our IVP we will just use our computer to generate a sequence of approximate numerical values for $x(t), y(t)$. This is the *numerical solution* of the problem produced using some *numerical method*.

As a prerequisite for developing the program for this project, you should then learn about numerical methods, and corresponding algorithms, to solve numerically (systems of) differential equations[1].

The idea at the basis of any algorithm for solving a given IVP is to rewrite the dx's and dt's

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f\left(x(t), t\right) \tag{7}$$

as finite steps δx and δt, and multiply the equation by δt. This gives algebraic relations to compute the change in the functions when the independent variable t is changed by δt. In the limit in which the stepsize δt is made very small, a good approximation to the underlying differential equation is achieved. So, and this is what we call the *Euler's method*, one discretizes the range $[t_0 = 0, T]$ over which t varies in steps of some given size $\delta t = h$ and iteratively determines the solution at all discrete times according to

$$x_{n+1} \approx x_n + hf(x_n, t_n) \tag{8}$$

advancing a solution from x_n to $x_{n=1} = x_n + h$, but neglecting, over the all t-range of integration, errors of order h. What the *second order Runge-Kutta* or *midpoint* method does is to attain better accuracy by combining the information from several Euler-style steps (each involving one evaluation of the function f). Second order accuracy is indeed achieved by using the initial derivative at each step to find a point halfway across the h interval, then using the midpoint derivative across the full width of the interval as summarized by the equations

$$\begin{aligned} k_1 &= hf\left(x_n, t_n\right) \\ k_2 &= hf\left(x_n + \frac{1}{2}k_1, t_n + \frac{1}{2}h\right) \\ x_{n+1} &= y_n + k_2 + \mathcal{O}(h^3). \end{aligned} \tag{9}$$

5 Finding the position of two consecutive maxima of the relevant numerically sampled functions

A good strategy to locate two consecutive maxima of some numerically sampled function (so in a vector) is to make use of numerical derivatives in a way that we now try to describe. We want to be able to state whether some t_i corresponds to a maximum in our numerically sampled $x(t)$. Then we can use the fact that we know $x(t_{i-1})$ as well as $x(t_{i+1})$ whose abscissas are $2h$ apart to approximate the derivative of $x(t)$ in t_i. With the same information we can build a numerical approximation for the second derivative. With the above derivatives at hand we can run over all sub ranges in $[0, T]$ and

- check whether the first derivative in correspondence to a given $[x_{i-1}, x_{i+1}]$ interval has the opposite sign with respect to the first derivative previously determined for the adjacent interval;
- check the sign of the second derivative in correspondence to the given interval.

These checks will allow you to generally find extrema of $x(t)$ and you can cook up an implementation that returns the position of two consecutive maxima. This way we will estimate the period of our periodic solutions $x(t)$ and $y(t)$. Moreover, we can estimate the lag (distance in t) between consecutive maxima of $x(t)$ and $y(t)$ telling us of how long the predators population lags behind the preys population as both populations fluctuate cyclically between their maximum and minimum values.

[1] You can, for instance, refer to http://en.wikipedia.org/wiki/Runge-Kutta_method or to §16.1 in "Numerical Recipes in C".

6 Numerically approximating the relevant integrals

Methods for the numerical integration of funcions of some single real variables are based on the naive strategy of summing up the value of the integrand at a sequence of abscissas within the range of integration[2]. We are interested in integrating our functions $x(t), y(t)$ whose value is known at equally spaced steps (once the corresponding differential equations are numerically solved). So, to introduce some notation we have a set of values of the independent variable $t_0, t_1, ..., t_N$ spaced apart by the constant stepsize h and functions $x(t), y(t)$ have known values at all t_i abscissas. What we want is to integrate $x(t), y(t)$ over the range $[t_0 = 0, t_N = T]$ as in Eq. (4). For this purpose you can use, at your choice,

- The *Trapezoidal rule*, which works by approximating the region under the graph of the function as a trapezoid and calculating its area

$$\int_{t_n}^{t_{n+1}} x(t)\mathrm{d}t \approx \frac{h}{2}\left(x_n + x_{n+1}\right) \tag{10}$$

- The *Simpson's rule*, where one replaces the integrand by the parabola which takes the same values as the integrand at the end points t_n and t_{n+2} and at the midpoint t_{n+1}.

$$\int_{t_n}^{t_{n+2}} x(t)\mathrm{d}t \approx \frac{h}{3}\left(x_n + 4x_{n+1} + x_{n+2}\right). \tag{11}$$

7 Putting things together, i.e. requirements for your program

In developing this project you are asked to:

(i) Write a program that integrates the system of ordinary differential equations of the basic predator-prey model Eq. (3), using a second order Runge Kutta integration scheme, to solve the corresponding IVP according to the strategy documented in Sec. 4. The user should provide in input, preferably via some input file, some/all of the following parameters[3]:

(a) For the definition of the model (all coefficients must be positive)
   (1) a as `preysReproductionCoefficient`
   (2) b as `preysMortalityCoefficient`
   (3) m as `predatorsMortalityCoefficient`
   (4) n as `predatorsReproductionCoefficient`

(b) For the Runge-Kutta scheme
   (1) The stepsize in your discretized time range `h`
   (2) The upper bound `tFinal` of your discretized time range
   (3) The time interval `tPrint`, in general different from the stepsize `h`, after which you want to append to your output file a new line containing the three values t $x(t)$ $y(t)$

(c) As boundary conditions for our IVP problem
   (1) $x(0)$ as `initialPreysPopulation`
   (2) $y(0)$ as `initialPredatorPopulation`

You can check your solution via plotting the results tabulated in the output files. For this task you can easily use gnuplot to produce the following plots

[2] You can, for instance, refer to https://en.wikipedia.org/wiki/Trapezoidal_rule and to https://en.wikipedia.org/wiki/Simpson%27s_rule

[3] Remember that choosing variables names is one of the decisions by which you can improve the readability of your code, along with choosing functions names.

(a) Plots of trajectories in the population-time plane, i.e. of $x(t)$ and $y(t)$ as functions of t;

(b) Plots of trajectories in the phase plane, i.e. of $y(t)$ as function of $x(t)$.

There are feature that should leap out from these first results, that we want to quantify numerically:

(a) The period T in the oscillations of $x(t)$ and $y(t)$ can be computed by finding the position of two consecutive e.g. maxima of the numerically sampled functions (see Sec. 5);

(b) The lag (distance in t) between consecutive e.g. maxima of $x(t)$ and $y(t)$;

(c) You will then use your result for T in order to compute averages of the two populations according to Eq. (4), using the techniques described in Sec. 6. At this point you will be able to check to which extent the properties in Eq. (4) are found to be satisfied in your numerical implementation of the model;

(d) The non-closed nature of trajectories in the phase plane if a too big value for δt is set (for this you will need to run the code multiple times). What you are expected to observe is that if δt becomes too large, then the approximations do not cycle around counterclockwise exactly to the initial point. To which kind of errors can this effect be ascribed?

(ii) Extend your program to solve the second version of our Lotka-Volterra IVP according to Eq. (5). The user, at this point, should be made able to decide at running time which model is he/she interested to, and coherently provide more input parameters to the program, including in particular the positive coefficient ϵ as `epsilon`.

(a) Repeat the same tasks you accomplished for the basic version of the model;

(b) Choose some fixed set of input values for all other parameters but `epsilon`, run your code at various `epsilon` values and monitor results for $\langle x \rangle$ and $\langle y \rangle$ (you can even make plots for this). How do your results explain the effects from the employment of DDT in citrus crops?

(iii) Finally, extend once more your program to solve the third and most realistic proposed version for our predator-prey IVP according to Eq. (6). The user, at this point, should be made able to decide at running time which model he/she is interested to, and coherently provide the necessary subset of input parameters to the program, including in particular the positive coefficients

(a) r as `preysCompetitionCoefficient`

(b) s as `predatorsCompetitionCoefficient`

At this point you should

(a) Check from your plots of trajectories in the population-time plane, i.e. of $x(t)$ and $y(t)$ as functions of t, that periodicity breaks down to some tendency towards equilibrium levels;

(b) Check from your plots of trajectories in the phase plane, i.e. of $y(t)$ as function of $x(t)$ that periodic motion is now replaced by motion toward an asymptotically stable rest point.

Once all of this is done, you will be mastering biological control techniques and able to switch to agriculture!

8 Code-related advices

- In the implementation of the second order Runge-Kutta scheme, a nice way to improve readability of your code is to define a *structure* storing in it all relevant quantities. Make sure you are familiar with `struct`s. Consider that, inside structures, you can also declare pointers to functions, which is useful given that the r.h.s. in our system of differential equation changes depending on the model we are addressing. A new type could be defined as follows

```
typedef struct RungeKutta2nd {
    int N;
    double * y;
    double * dy;
    double ys;
    void (*stepRK2)(double, double);
    void (*rhsBasicModel)(double, double *, double *);
    void (*rhsDdtModel)(double, double *, double *);
    void (*rhsCompetitiveModel)(double, double *, double *);
} RungeKutta2nd;
```

On top you will have a function to initialize the new type like

```
void Initialize_RK2(RungeKutta2nd *rk2, /*...*/);
```

or, gathering input from an input file, something like

```
void Initialize_RK2(RungeKutta2nd *rk2, char *filename);
```

In such a function you will have to assign the function pointers to the corresponding functions like

```
rk2->rhsBasicModel = _rhsBasicModel;
rk2->rhsDdtModel = NULL;
rk2->rhsCompetitiveModel = NULL;
```

if you are initializing `rk2` for the solution of the basic model and, analogously for the other two cases.

- Something that you will need to repeatedly do in the flow of your program is to print data to the output file. Saving data too frequently to your output file will slow down your program without adding much information to the final outcome. This is the reason for the already introduced `tPrint` variable. Moreover, it might be a good idea to add to your output file also a headline to make clear what information does each column contains. If you use gnuplot, lines starting by `#` in the output files will be automatically ignored.

- Finally, the following list of (incomplete) function signatures for further functions you need to implement in your code is provided as a guideline to help you envision how to organize your implementation.

```
/*type*/ calcConsecutiveMaxInVector(/*...*/);
/*type*/ calcPeriodsOfOscillations(/*...*/);
/*type*/ calcLagTime(/*...*/);
/*type*/ calcIntegralWithTrapeziRule(/*...*/);   //otherwise
/*type*/ calcIntegralWithSimpsonRule(/*...*/);
/*type*/ calcRatiosOfCoefficientsAndCompareWithAvgPopulation(/*...*/);
```

Thankfully,

a Californian citrus crop...