Final project

Lotka-Volterra model

The simple model (i)

- Assume two animal species: a predator (x) and its prey (y) (e.g. foxes and rabbits)
- Imagine that there is no prey. Clearly the predators will extinguish because there is not food for them. It is also obvious that the speed of disappearance will increase with the number of individuals. Mathematically:

$$\frac{dx}{dt} = -\alpha\, x$$

The simple model (ii)

- Conversely, as more preys available, the more the population of predators increases. This increase should be proportional to the number of contact predator-prey, which in turn depends on the product of the respective populations.
- This effect modifies the previous equation. Then

$$\boxed{\frac{dx}{dt} = -\alpha\, x + \beta\, x\, y}$$

The simple model (iii)

- Let’s focus now on the prey. By the moment being, assume that the supply of vegetables is unlimited
- If there were no predator, the prey population would grow exponentially:

$$\frac{dy}{dt} = \kappa\, y$$

- But the effect of the predators will be the opposite to the one before:

$$\boxed{\frac{dy}{dt} = \kappa\, y - \lambda\, x\, y}$$

The simple model (iv)

- The two framed equations constitutes the Lotka-Volterra model

$$\frac{dx}{dt} = -\alpha\, x + \beta\, x\, y$$

$$\frac{dy}{dt} = \kappa\, y - \lambda\, x\, y$$

- Despite the equations look rather simple, there exists no analytical solution
- Numerical methods are thus required

The logistic model (i)

- Assuming that food supply is unlimited for the prey is not realistic
- Instead, it is better to expect that some kind of equilibrium will be reached, as the population –even in the absence of predators– cannot grow indefinitely.
- This can be simulated by adding a term proportional to the square of the population to the previous model:

$$\frac{dy}{dt} = \kappa\, y - \kappa' y^2 - \lambda\, x\, y$$

The logistic model (ii)

- As before, an analogous modification must be applied to the other species. The full extended model, then, reads:

$$\frac{dx}{dt} = -\alpha\, x + \alpha' x^2 + \beta\, x\, y$$

$$\frac{dy}{dt} = \kappa\, y - \kappa' y^2 - \lambda\, x\, y$$

- Of course, again the solution must be found in a numerical form. (Unless λ and β equals 0)

Numerical solution of differential equations

Taylor's method

- Taylor's series expansion of a function $y(t)$ around $t = t_0$ is given by

$$
\begin{aligned}
y(t) &= y(t_0) + \frac{1}{1!}y'(t_0)\,(t-t_0) + \frac{1}{2!}y''(t_0)(t-t_0)^2 \\
&\quad + \frac{1}{3!}y'''(t_0)(t-t_0)^3 + \cdots \\
&= y_0 + y_0'\,h + \frac{1}{2}\,y_0''\,h^2 + \frac{1}{2}\,y_0'''\,h^3 + \cdots
\end{aligned}
$$

- Then to approximate $y(t)$ numerically for the initial value problem $\{y'(t) = f(t,y); y(t_0) = y_0\}$, we simply substitute the values of y_0 and its successive derivatives in the Taylor expansion before.

- But this can be very computationally expensive, especially when t is far from t_0.

Euler's method

Maybe the easiest method is to truncate Taylor's expansion at first order

$$y_1 = y(t_1) = y(t_0 + h) \approx y_0 + y_0' \, h = y_0 + h \, f(t_0, y_0)$$

Then

$$y_2 = y(t_2) = y(t_1 + h) \approx y_1 + h \, f(t_1, y_1)$$

and so on. An iterative method is, therefore, trivial

$$y_{n+1} \approx y_n + h \, f(t_n, y_n)$$

Obviously, h must be kept small to avoid divergences. Therefore the method is rather slow and needs to be improved

Modified Euler's method

It is evidently true that $y_1 = y_0 + \int_{t_0}^{t_1} y'(t)\,dt = y_0 + \int_{t_0}^{t_1} f(t, y)\,dt$

So, looking from a different perspective, what Euler's method does is to approximate the integral by keeping $f(t, y) \approx f(t_0, y_0)$ constant:

$$y_1 \approx y_1^{(0)} = y_0 + h\,f(t_0, y_0)$$

A better approximation can be achieved by taking (trapezoid method)

$$y_1^{(1)} = y_0 + \frac{h}{2}\left[f(t_0, y_0) + f\left(t_1, y_1^{(0)}\right)\right]$$

Now, the approximation can be successively improved until convergence

$$y_1 = y_1^{(n+1)} = y_0 + \frac{h}{2}\left[f(t_0, y_0) + f\left(t_1, y_1^{(n)}\right)\right]$$

Finally, repeat the same procedure for $y_2, y_3, \ldots\ y_n$

Runge – Kutta's method (i)

- Runge – Kutta's method is a more elaborated development based on the same concepts that Euler's and modified Euler's methods.
- We have already seen that Euler's method is a first order Taylor method.
- Moreover, it can also be proven that modified Euler's method is a second order Taylor method

Runge – Kutta's method (ii)

- Let's write the first iteration of modified Euler's method in a different way

$$y_1 = y_0 + \frac{h}{2}\left[f(t_0, y_0) + f\left(t_1, {y_1}^{(0)}\right)\right]$$
$$= y_0 + \frac{h}{2}\left[f(t_0, y_0) + f\big(t_0 + h, y_0 + h\, f(t_0, y_0)\big)\right]$$
$$= y_0 + \frac{1}{2}\left[K_1 + K_2\right]$$

with $K_1 = h\, f(t_0, y_0)$ and $K_2 = h\, f(t_0 + h, y_0 + K_1)$

- Obviously, in Euler's method $y_1 = y_0 + K_1$

Runge – Kutta’s method (iii)

- Basically, Runge – Kutta’s method generalizes the previous observation to attain higher-order Taylor approximations.
- The most used is fourth-order Runge – Kutta’s method, which is a fourth-order Taylor approximation and that can be written as

$$y_1 = y_0 + \frac{1}{6}\,[K_1 + 2\,K_2 + 2\,K_3 + K_4\,]$$

with

$$K_1 = h\,f(t_0, y_0) \qquad K_2 = h\,f\left(t_0 + \frac{h}{2}, y_0 + \frac{K_1}{2}\right)$$

$$K_3 = h\,f\left(t_0 + \frac{h}{2}, y_0 + \frac{K_2}{2}\right) \qquad K_4 = h\,f(t_0 + h, y_0 + K_3)$$

The exercise

The program (i)

The goal of the exercise is to code a Fortran program that solves the two discussed models for the population growth in the predator-prey system.

The program must ask for the model to solve for and the numerical method to be used. Furthermore, there should be an option for computing the error of a particular method with respect to fourth-order Runge-Kutta’s, taken as reference.

The time-length of the simulation, constants $(\alpha, \alpha', \beta \ldots)$ and initial populations as well as the step (h) must be given in a separate file. Referring the step, a possible improvement could be that the program suggests it by solving the model equation(s) for the prey with $\lambda = 0$ and comparing to the exact solution in this case

The program (ii)

The solutions should be kept in an appropriate derived-type variable.

The output must consists in at least two files: a formatted one with the results conveniently sorted and printed in tabular form and as many as needed unformatted files consisting each of them of the set of points $(t_i \quad x_i \quad y_i)$ for each method and model required in input.

Finally, a makefile to compile and link the source file(s) is also required

The minimum requisite to pass is that the programs works, but the use of more elaborated structures (subprograms, modules…) will be also graded.