Simulation of a wolf population by the E-10-Simulation GmbH

Lars Schiller

April 23, 2015

Contents

1 Introduction and task description

In 2011, wolves where sighted in Luneburg Heath for the first time (after quite a long period). The E-10-Simulation GmbH was asked by the Ministry of Food, Agriculture and Consumerism of Lower Saxony to forecast the development of the wolf population over the next 15 years.

The E-10-Simulation GmbH asks all interested students to submit a simulation proposal. The main task consists of modelling the development of the population mathematically, choosing a suitable solver for the simulation, writing a corresponding software program and evaluating the results. Further technical details and specifications are listed in the attachment.

2 Mathematical model of the wolf population

From the Task Sheet it is known, that the population size of the wolves in Luneburg Heath in the year 2011 is equal to 20. This yields to equation 1:

$$w(t_0) = w_0 = 20 \qquad , \quad t_0 = 2011. \tag{1}$$

Furthermore it is known, that the growth rate of a wolf population is proportional to the current population with respect to a constant factor. Due to a limited amount of food, the growth rate will be also proportional to $1 - b * w$. This yields to equation 2 and equation 3:

$$\dot{w} \sim a \cdot w \qquad , \quad a = 0.055, \tag{2}$$

$$\dot{w} \sim 1 - b \cdot w \qquad , \quad b = \frac{1}{250}. \tag{3}$$

In order to obtain a differential equation which describes the behaviour of the wolf population adequate, and therefore fulfils equation 2 and 3, one can simply multiply the both restrictions, which

corresponds with a logical `AND`-connector. This yields to the following ordinary differential equation (ODE):

$$\dot{w} = (aw)\,(1 - bw)\,. \tag{4}$$

In the following section this ODE will be solved by different numerical methods.

3 Solutions of the mathematical model obtained by different numerical methods

The following different numerical methods are used to solve the ODE:

1. the Simple *Euler*-Method,
2. the advanced *Euler*-Method and
3. the *Runge-Kutta*-Method.

3.1 Simple *Euler*-Method

This is the most simple method to solve an ODE with initial condition. It is based on the *Taylor*-series expansion. The *Taylor*-series expansion says, a function $f(x)$ can be described around a working point x_0 as the sum of its derivatives, with the following rule

$$Tf(x)\Big|_{x_0} = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \tag{5}$$

$$= f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{1}{2}f^{(2)}(x_0)(x - x_0)^2 + \cdots + \frac{1}{\infty!}f^{(\infty)}(x_0)(x - x_0)^\infty, \tag{6}$$

where $f^{(n)}$ denotes the n-th derivative of f with respect to x. For an initial value problem with the form

$$y' = f(x, y) \quad , \quad y(x_0) = y_0$$

the linearised (only the first derivative is respected) and discretized version of equation 6 is the following:

$$y_{n+1} = y_n + \frac{\delta}{\delta x}y(x_n) \cdot \underbrace{(x_{n+1} - x_n)}_{\text{step size: } h_n} = y_n + f(x_n, y_n) \cdot h_n. \tag{7}$$

Since the first derivative of f is given and we also know the initial value, equation 7 can be used within a `while`-loop to calculate the values of the desired function. The only thing we don't know is the step size. The most simple way is to set the step size to a constant value:

$$h_n = h_0.$$

The solution depends strongly on this step size h_0. In figure 2 the blue curve represents the result of this method.

3.2 Advanced *Euler*-Method

The main idea of this method is equal to the simple *Euler*-Method. Based on the linearised *Taylor* series expansion, the solution is calculated recursively. But this time we change the step size individually, with respect to the changes in the right hand side of the ode. Therefore the changes of step size will be proportional to the second derivative of the desired function. In every calculation step the next value of the solution is calculated with the current step size (like the simple *Euler*-method) and stored in y_{n+1}. Furthermore the its calculated by doing two steps with half the current step size (like doing the simple *Euler* twice) and stored in $\tilde{y}_{n+1}$. The difference of these two values is an good estimator for the local error we do. Therefore the local error estimator φ is defined:

$$\varphi(x_n, y_n, h_n) = 2(\tilde{y}_{n+1} - y_{n+1}) \tag{8}$$

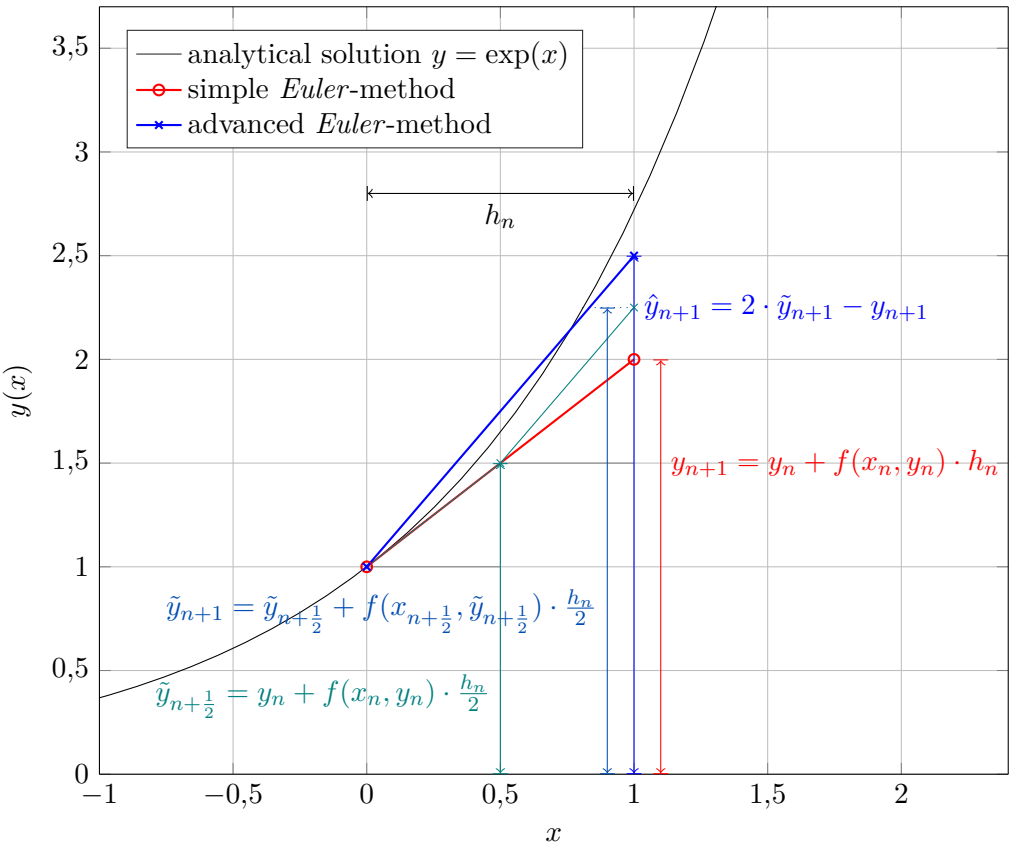

Figure 1: principle scheme of the advanced *Euler*-Method by means of the initial value problem: $\dot{y} = y, y(0) = 1$

If the local error is great the step size decreases, and if its small the step size increases.

Another important feature of the advanced *Euler*-method is the usage of a linear combination of the both calculated values (y_{n+1} and $\tilde{y}_{n+1}$) as the next value of the solution $\hat{y}_{n+1}$. Therefore the order of the method is increased. The local error of the simple *Euler*-method is a function of h_n^2. In comparison to that the advanced method holds a local error $\hat{\varepsilon}_n = \mathcal{O}(h_n^3)$. The mathematical explanation for this can be found in the lecture Script on page 15ff. The principle scheme of the advanced *Euler*-method is represent in figure 1 and the solution of the given problem calculated with this method is represented by the red curve in figure 2.

3.3 *Runge-Kutta*-Method

This solver is implemented by the Mathwork guys. I hope to understand the mathematics of `ode45` within this lecture. The result of this solver is represented by the green curve in figure 2 and will be used as an reference for the exact solution in the evaluation.

4 Description and evaluation of the results

The results of all solvers are reasonable. First the wolf population grows slowly and than faster and faster, which corresponds perfectly with the constraint given in equation 2. When the upper limit of the population (250) is almost reached, the slope begins to decrease, which is demand by the constraint given in equation 3.

In table 1 some evaluation criteria are listed. The calculation time of the simple *Euler*-method and the advanced one are nearly the same. In figure 2 one can see that the solution of advanced *Euler* almost perfectly fits the one of `ode45`, which can be treated as exact solution. The solution of the simple method is not that exact. Its too inert to realize the changes. Furthermore the advanced

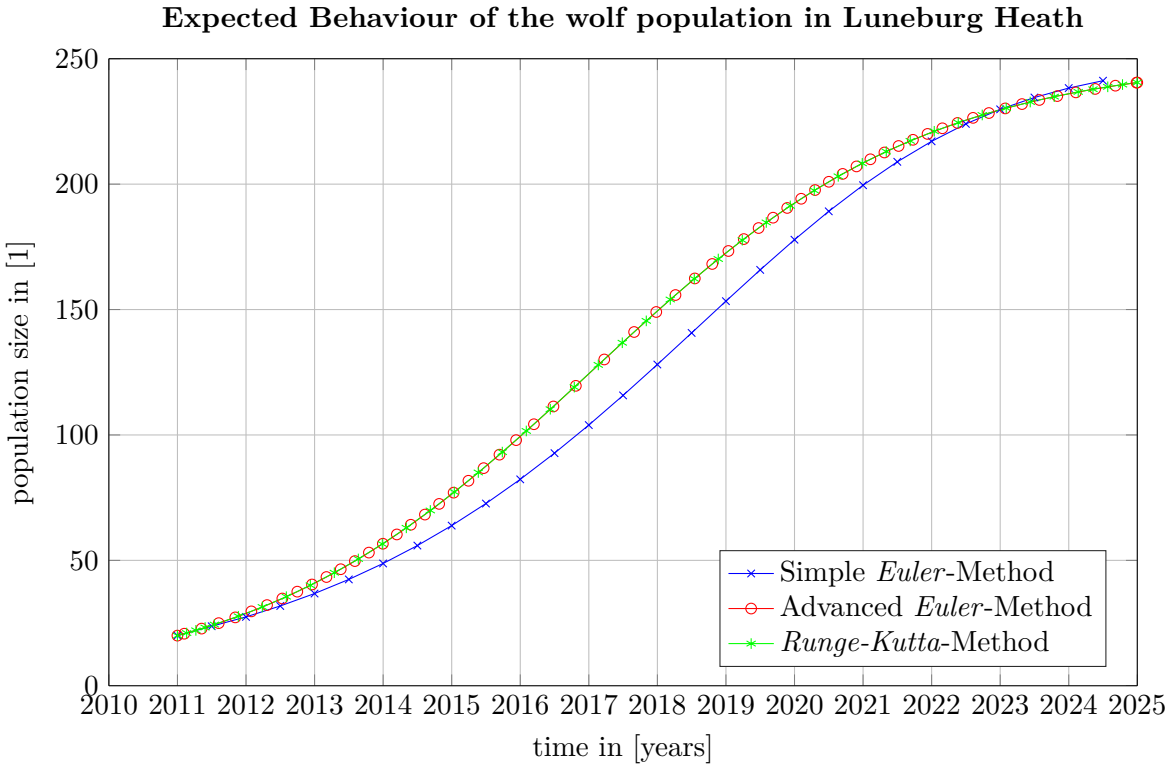

Figure 2: Solutions of the ODE (step size of simple *Euler*-method: $h_0 = .5$)

criteria	Simple *Euler* ($h_0 = .5$)	Advanced *Euler*	`ode45`
supporting points	29	21	45
calculation time [sec]	0.0057	0.0061	0.1710
global error	-2.97	0.20	(reference)

Table 1: Evaluation of the results

method needs less supporting points than the simple method. Therefore the storage space which is needed will be much smaller.

In summary one can say the advanced *Euler*-method gives much more qualitative results, needs less storage space and the computational effort is at least for this case not really greater than the one of the simple method.