Tutorial Numerical ODE solving

Essential Skills for Bioinformatics and Biocomplexity

Topics covered:

- Different algorithms for numerical ODE solving.
- Common problems in numerical ODE solving.

1 Numerical ODE solving

For this exercise we will work in the RStudio environment. To do so, first make sure you have installed R and RStudio on your device. Since we will be solving ordinary differential equations we will use the R package `deSolve` developed by Karline Soetaert and colleagues[**?**]. To install this package open RStudio, go to the Tools menu, choose Install Packages and then type `deSolve`.

Question 1. Consider the following differential equation:

$$\frac{dx}{dt} = x(1-x)$$

We will try to numerically solve this differential equation using simple forward Euler integration.However, this differential equation has an analytical solution, that we can compare our numerical solutions to:

$$x(t) = 0.1\frac{e^t}{1+0.1(e^t-1)}$$

a. Consider the code given in BasicNumericalIntegration.Rmd. Run the code and make sure you understand what is happening in each line

b. Modify the code to compute the overall summed difference between analytical and numerical solution. For this you will need to define a new variable, summeddif, and add for each integration time step the difference between the two solutions, store it in this variable, and print out its final value.

c. Play with the duration of the integration timestep. What happens to the goodness of fit between numerical and analytical solution? Do you see strange things happening?

Let us consider a population of bacteria growing on a resource. We assume that the resource consumption per bacterium is a saturating function, such that the maximum rate at which

bacteria can consume resource is limited. Following Levin *et al.* (2013), we model the resource concentration, R, and the number of bacteria, B, using two differential equations:

$$\frac{\mathrm{d}R}{\mathrm{d}t} = -e\frac{vR}{k+R}B \qquad \frac{\mathrm{d}B}{\mathrm{d}t} = \frac{vR}{k+R}B. \tag{1}$$

The parameters and initial conditions estimated by Levin *et al.* are: $v = 1.4\,\mathrm{h}^{-1}$, $e = 5\times10^{-7}\,\mu\mathrm{g}$ per cell, $k = 1\,\mu\mathrm{g/mL}$, $R(0) = 350\,\mu\mathrm{g/mL}$. Furthermore, choose $B(0) = 10^3$ cells.

Question 2.

a. Make a copy of the script BasicNumericalIntegration.Rmd from the previous exercise, give it another name and use it as a starting point for this exercise. Now modify this script to compute the numerical integration of the system of two equations provided above.
A few hints:
1) Note that in these equations there are parameters, values of these parameters need to be defined at the top of the script.
2) Note that both equations need to be described as functions, as before with `function(x)...x...` now they are functions of both variables, so use `function(R,B)...R...B`

b. Run your code and explain what happens to R and B. What does this imply?

c. Study what happens if you change the time step used by the Forward Euler method. Explain your results.

d. What is the maximal time step that still yields stabble and biogically plausible results?

e. There are many different algorithms for solving ODEs that are more complicated than the Forward Euler Method, but also less prone to integration errors. The script `Q2_EulerVersusRK.Rmd` compares the Forward Euler Method to another method, RK23, which uses a Runge-Kutta pair of second and third order, using the R-package `deSolve`. The RK23 algorithm uses a *dynamical time step*. Familiarise yourself with the content of the script, and use it to compare both methods for different values of the time step. Explain your results.

f. Another way of preventing integration problems is by altering the function that is integrated. Look at the line in the model definition that is currently commented out. Explain how this line could help, and investigate to what extent this is indeed the case.