NSC1002 Mathematics and Computing: Integrative Tools for Natural Sciences

Dr. Blanca Ayarzagüena and Beth Wingate

University of Exeter

By the end of this lecture you will…

- know how to solve ordinary differential equations numerically.

- learn explicit and implicit methods.

Ordinary differential equations

- You know what ordinary differential equations (ODEs) are:

$$y'(y,t) = f(y,t) \qquad y(0) = y_0$$

- Also can be written

$$\frac{dy}{dt} = f(y,t)$$

- Example:

$$y'(t) = y(t) \qquad y(0) = 3$$

  Solution:

$$y(t) = 3e^{t}$$

- This is a very simple example but there are many others more complicated and solving them by hand is really though or we don't even know the analytic solution.

Numerical methods can make it easier (and so, your life)!!

Numerical methods for solving ODEs: **Forward Euler or Explicit Euler** (one-step)

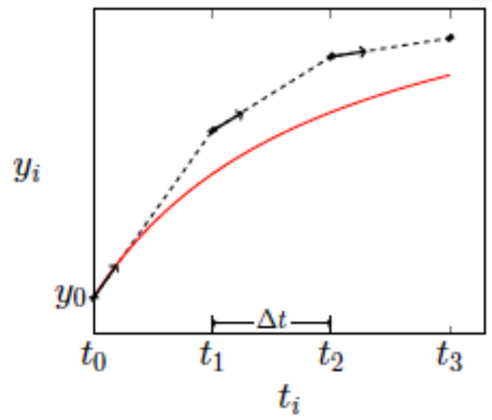
- We come back to the general ODE expression:

$$\frac{dy}{dt} = f(y,t) \qquad y(0) = y_0$$

- We can replace the analytical derivative with the numerical one:

$$\frac{y_{i+1} - y_i}{\Delta t} = f(y_i, t_i) \qquad\qquad y(0) = y_0$$

$$y_{i+1} = y_i + \Delta t\, f(y_i, t_i)$$



And if we know a point in time and space (i=0), we can start iterating by our time-step to find the value of y in the next point (i+1).

Numerical methods for solving ODEs: **Forward Euler or Explicit Euler** (one-step)

$$y_{i+1} = y_i + \Delta t\, f\left(y_i, t_i\right)$$

```
import numpy as np

y[0] = y0    # You should initialise y earlier
t = t0
# Loop where we compute the value of y for each point
for i in range(1, npoints):
    # you should initialise df1 too
    y[i] = y[i-1] + timestep * df1(t[i-1], y[i-1])
    t +=  timestep
```

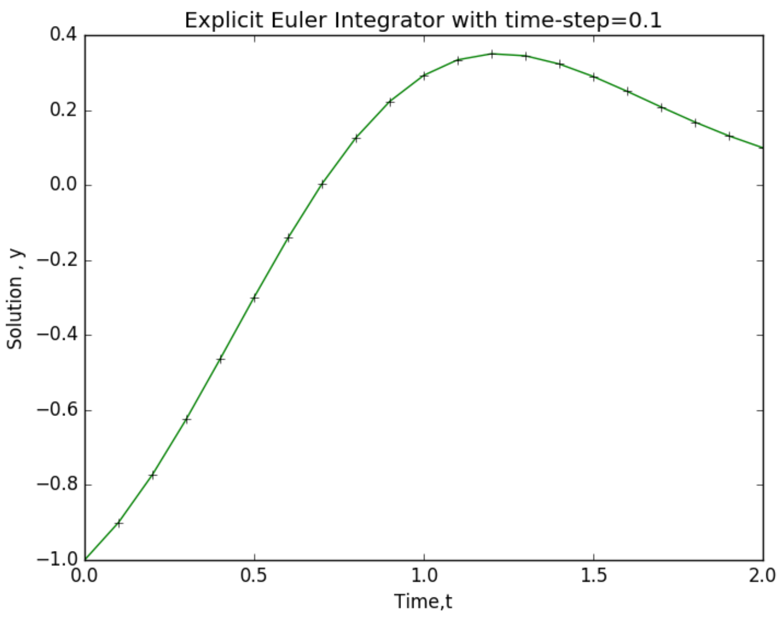
This is only a help for coding. You have to initialize and/or define many variables beforehand!!!!

Numerical methods for solving ODEs: **Forward Euler or Explicit Euler** (one-step)

Example:

$$\frac{dy}{dt} = e^{t-t^2} - 2\,t\,y \qquad\qquad y(0) = -1$$

$$time - step = 0.1$$



Numerical methods for solving ODEs: **Adams-Bashforth 3rd order** (linear multistep)

- Linear multistep methods: they use a combination of solutions from several previous timesteps (instead of only one). They usually provide a better approximation of the ODE solution than one-step methods.
- An example of a linear multistep method is **Adams-Bashforth 3rd order** (also explicit).
- This is the expression:

$$y_i = y_{i-1} + \frac{\Delta t}{12}\left[23 f\left(y_{i-1}, t_{i-1}\right) - 16 f\left(y_{i-2}, t_{i-2}\right) + 5 f\left(y_{i-3}, t_{i-3}\right)\right]$$

Numerical methods for solving ODEs: **Adams-Bashforth 3rd order** (linear multistep)

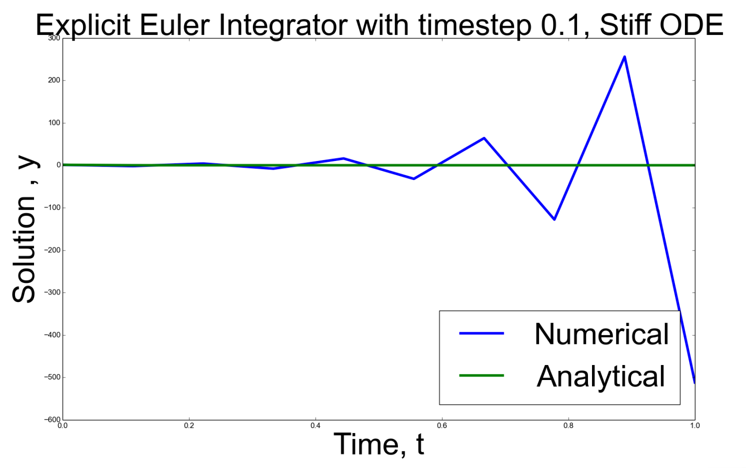
$$y_i = y_{i-1} + \frac{\Delta t}{12}\left[23 f\left(y_{i-1}, t_{i-1}\right) - 16 f\left(y_{i-2}, t_{i-2}\right) + 5 f\left(y_{i-3}, t_{i-3}\right)\right]$$

- *Problem*: you need to know at least the solution in 3 previous points and probably you only know it in 1.
- *Solution*: Use other methods to fill in the gaps: lower-order Adams Bashforth methods, **explicit Euler method**…

You will use this!

Numerical methods for solving ODEs: **explicit vs implicit method**

- Some ODEs do not work well with explicit methods (stiff ODEs).
- You usually get stupid solution for large time-steps.



- Methods that usually avoid this problem: **implicit methods**. They find the solution based on the current and later state.

Numerical methods for ODEs: **Implicit Euler** (one-step)

- The expression is:

$$y_{i+1} = y_i + \Delta t f\left(y_{i+1}, t_{i+1}\right)$$

You have now y_{i+1} on both sides of the equations, so you would need a method to solve the non-linear equation (such as Newton-Raphson)

Now…

Let’s practice what we have learnt today with the worksheet exercises.