Assignment 3

Ayden S. McCann

March 2021

Code Listing at the end

Introduction

The purpose of this assignment is to test implement and analyse various methods for first, numerical differentiation, and then using this knowledge implement some methods of solving initial value problems and finally (and optionally) apply this to a Lotka-Volterra system using the scipy solve_ivp function.

1 Question 1

1.3

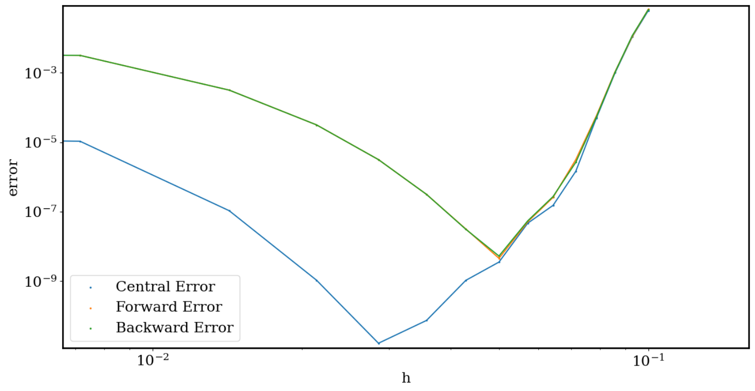

Figure 1: The errors for changing step sizes (h). Range cropped to better highlight changing behaviour for larger values of h.

1.4

As can be seen in Figure 1, both the forward and backward methods share almost identical error. Both of these methods are a first-order approximation (error is $O(h)$). There is a point where the error begins increasing for decreasing values of h. This turn off point happens at a later stage for the central error method, which scales $O(h^2)$

1.5

At increasingly small values of h the subtraction utilised in all 3 methods produces a large rounding error due to the errors being stored as numpy.float64 numbers, which have a finite precision of 8 bytes.

2 Question 2

$$N'(t) = -\frac{1}{\tau}N(t) \tag{1}$$

Using: $\tau = 5$, $t_{max} = 25$, $N(0) = 100$, $\Delta t = 0.5$

2.1

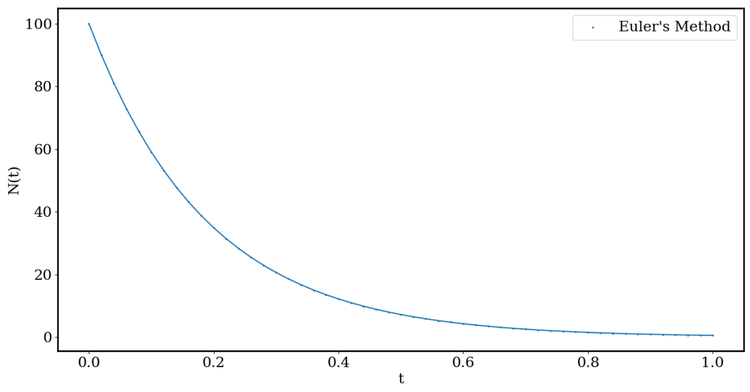

Figure 2: Euler Method of (1) using the above parameters

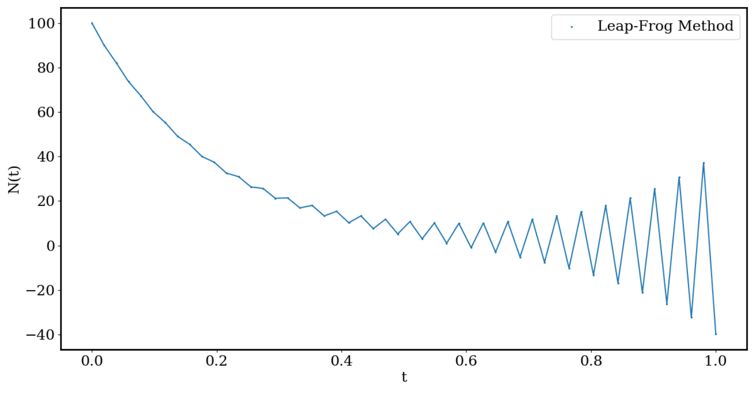

Figure 3: Leap-Frog Method of (1) using the above parameters

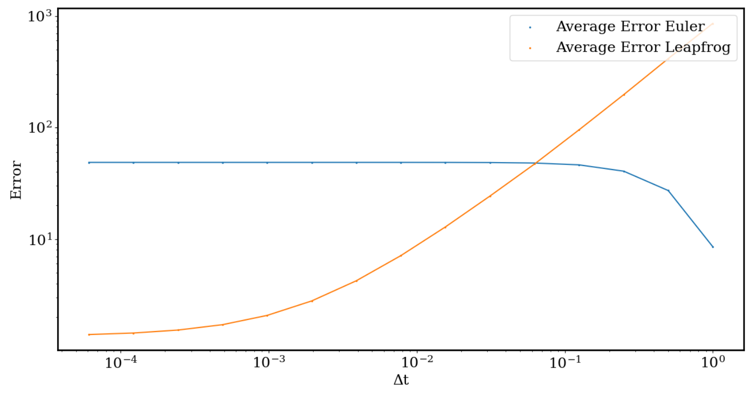

Figure 4: The errors for changing step sizes Δt. Range cropped to better highlight changing behaviour for larger values of Δt.

2.4

As can be seen in Figure 4, the Euler method is superior for $\Delta t > 6.37 * 10^{-2}$. The error from the Euler method increases until it reaches an approx value of 48 as Δt approaches zero. For the Leap-Frog method however, the error steadily decreases and becomes more and more accurate with decreasing Δt.

3 Question 3

As can be seen below in Figure 5 (Linear) and Figure 6 (Log-Log) the error for the Runge-Kutta method of solving the initial value problem increases drastically above ~ 0.6. Although this method is second order and global error is O(Δx^2), visual inspection the large-scale error can be approximated to scale as $\propto 1200\, x^8$ (Best demonstrated in Figure 5). However, as seen in Figure 6, this does not match the behaviour at low values of Δt. The Runge-Kutta error tends to ~ 1.47 as $\Delta t \to 0$ as opposed to zero like $1200x^8$

***** note to marker *****

I've only realised with less than an hour til the due date the question asked specifically for $t = 1$, while i have summed over all timesteps in the range $[0, 1]$.

I've included my quick attempt at rectifying this as 3_fixed.py but i haven't had time to verify if it is correct.

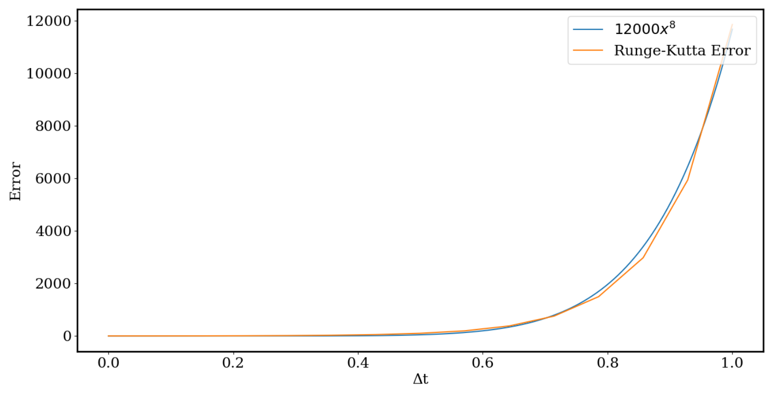

Figure 5: The errors for changing step sizes Δt. (Linear axes)

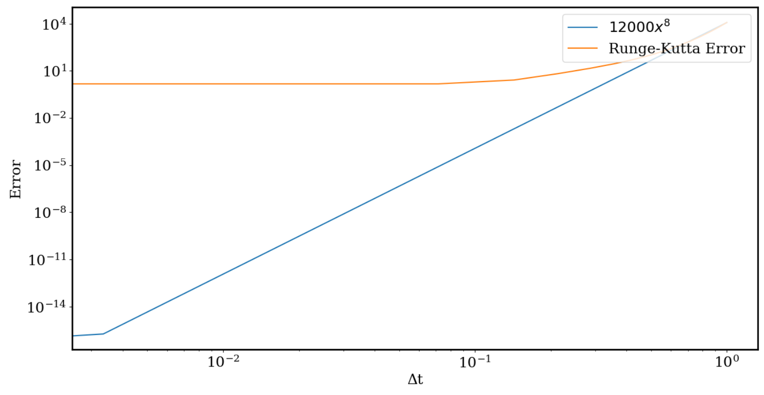

Figure 6: The errors for changing step sizes Δt. (Log-Log axes)

4 Question 4

4.2

The algorithm used by solve_ivp is (by default) an Explicit Runge-Kutta method of order 5(4) also known as RK45. The documentation describes the error of this method as ”controlled assuming accuracy of the fourth-order method, but steps are taken using the fifth-order accurate formula (local extrapolation is done)”.

4.3

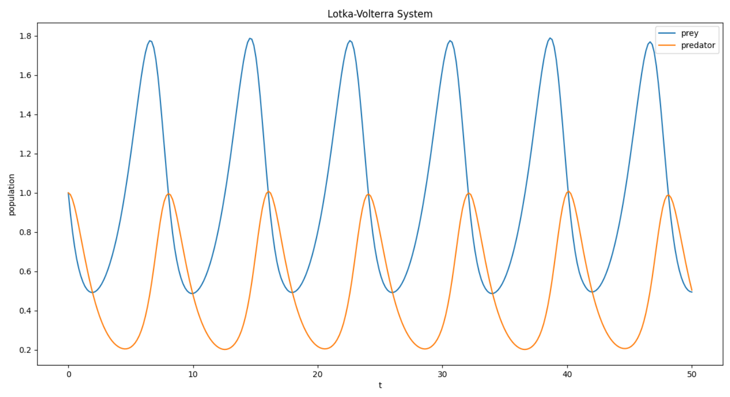

Figure 7: The development of the two populations (predators and prey) over time.

4.4

Equilibrium value (x,y) = (1, 0.5)

4.5

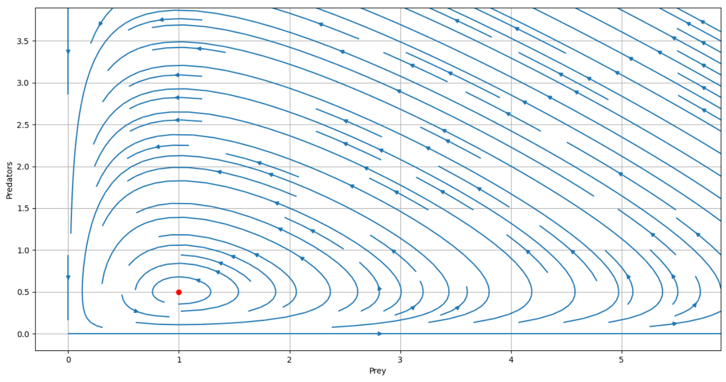

Figure 8: The phase portrait showing the development of the system given certain initial conditions. Not that most (if not all) trajectories within the range of this Figure appear to be stable cycles around the fixed point at (1, 0.5)

Discussion

As demonstrated in this assignment there are various methods for numerically obtaining the gradient at a given point of a function. As seen in Figure 1 the order of this approximation governs the scaling of the error and as a result deems certain methods more useful than others in particular circumstances. There are also several methods for solving initial value problems such as Question 2 where the Leap-Frog and the Euler method's errors are compared, when working with a certain value of Δt it is important to recognise the errors a given IVP method will produce, as seen in Figure 4 for instance, the Euler method is more accurate for part of the range of Δt and the Leap-Frog method is more accurate for another part.

5 Code Listing

Question 1.1 - 1.1.py

Question 1.2, 1.3 - 1.2.py

Question 1.4, 1.5 - text based, in pdf

Question 2.1 - 2.1.py

Question 2.2 - 2.2.py

Question 2.3 - 2.3.py

Question 2.4 - text based, in pdf

Question 3.1, 3.2 - 3.py and 3_fixed.py as discussed earlier

Question 4.1, 4.3, 4.4, 4.5 - 4.py

Question 4.2 - text based, in pdf