Exercise # 1. Numerical methods for ODES.

Alexandre Rodrigues (2039952)

November 25, 2021

Question 1

The Simpson's method is a method to solve Ordinary Differential Equations (ODEs) defined by

$$y_{n+2} = y_n + \frac{h}{3}\left(f_n + 4f_{n+1} + f_{n+2}\right). \tag{1}$$

The ODE to solve in this question is

$$y'(t) = -5y(t) = f(t_n, y_n), \qquad y(0) = 1. \tag{2}$$

To solve this ODE with the Simpson's method the value of $y_1 = y(h)$ also needs to be determined.

One approach is to use the Forward Euler method, in the following way:

$$y_1 = y(t = h) = y(0) + hf(0, y(0)) \tag{3}$$

Another method one can use is the 4-th order Runge Kutta method, given by

$$k_1 = f(0, y(0)) \tag{4}$$
$$k_2 = f\left(\frac{h}{2}, y(0) + \frac{h}{2}k_1\right) \tag{5}$$
$$k_3 = f\left(\frac{h}{2}, y(0) + \frac{h}{2}k_2\right) \tag{6}$$
$$k_4 = f\left(h, y(0) + hk_3\right) \tag{7}$$
$$y_1 = y(t = h) = y(0) + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{8}$$

The results of the computations were then compared with the exact solution

$$y(t) = e^{-5t}, \tag{9}$$

to understand the influence of the methods used to compute $y(t = h)$.

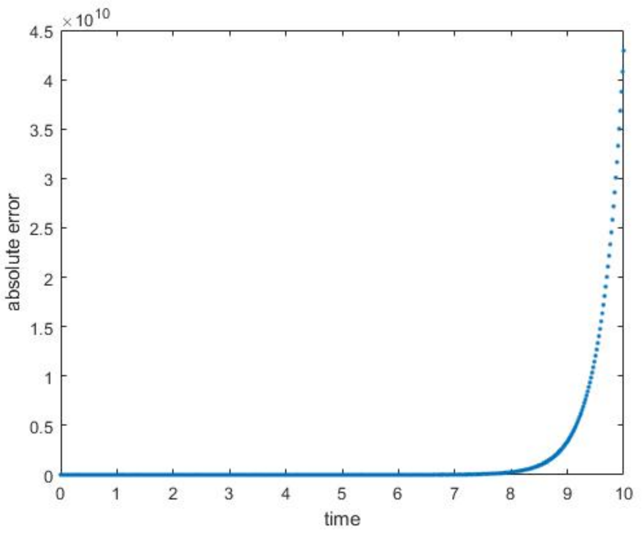

Figure 1: Absolute error in function of time using the Forward Euler method to compute $y(t = h)$

The final error when using the Forward Euler to compute $y(t = h)$ is 4.2916×10^{10}.

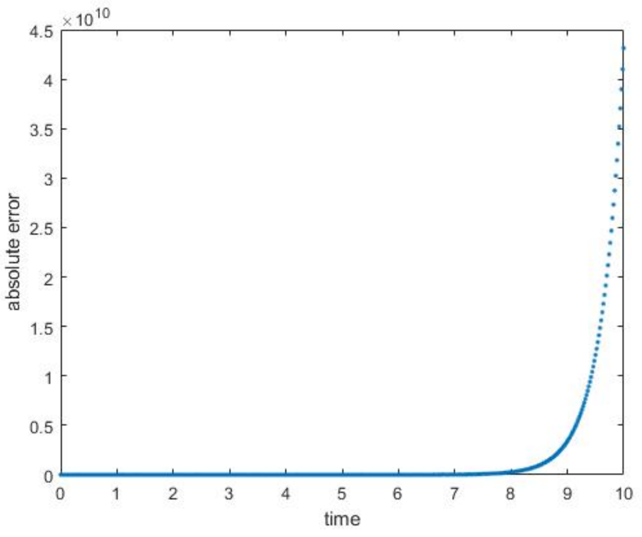

Figure 2: Absolute error in function of time using the RK4 method to compute $y(t = h)$

The final error when using the 4-th order Runge Kutta method to compute $y(t = h)$ is 4.3146×10^{10}.

The Simpson's method has an empty stability region, which explains the large final error. The FE calculation for $y(t = h)$ is better then the RK4 calculation given the lower final error. This is, although, not very relevant because the difference is about $0.5 \times 10^{-10}\%$.

Question 2

The method used for this question is the 4-th order Runge Kutta method defined by

$$k_1 = f(t_n, y_n) \tag{10}$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \tag{11}$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \tag{12}$$

$$k_4 = f\left(t_n + h, y_n + hk_3\right) \tag{13}$$

$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{14}$$

The Ordinary Differential Equation to solve in this question is

$$y'(t) = -10y^2(t) = f(t_n, y_n), \qquad y(0) = 1. \tag{15}$$

The exact solution is

$$y(t) = \frac{1}{10t+1}, \tag{16}$$

which will be used to compute the error as the difference between the exact and the experimental solutions.

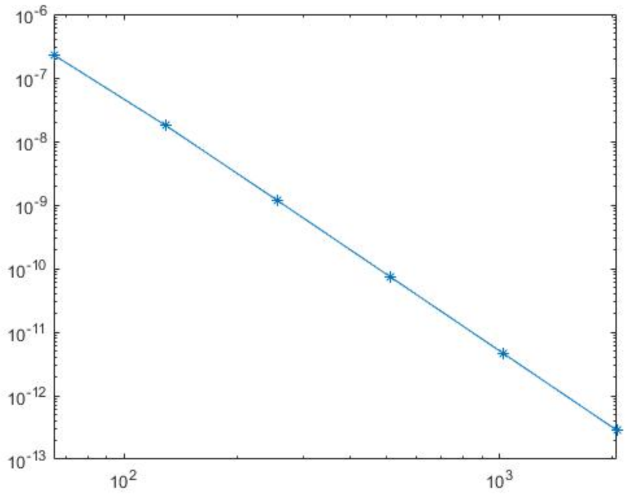

Figure 3: LogLog plot of the error as a function of the number of steps.

h	**error**
3.125000×10^{-2}	2.291844×10^{-7}
1.562500×10^{-2}	1.785763×10^{-8}
7.812500×10^{-3}	1.160234×10^{-9}
3.906250×10^{-3}	7.312862×10^{-11}
1.953125×10^{-3}	4.579586×10^{-12}
9.765625×10^{-4}	2.863750×10^{-13}

Table 1: Final error for each value of h

In figure 3 and table 1, you can see the clear decrease in the final error with the increase of the number of steps (halving h) as expected in theory.

Question 3

BDF2 derivation

The usual form of the ODE:

$$y'(t) = f(t, y(t)) \tag{17}$$
$$y(t_0) = y_0 \tag{18}$$

Writing this ODE on a point t_{n+k}:

$$y'(t_{n+k}) = f(t_{n+k}, y_{n+k}) \tag{19}$$

The general expression of the Backward Differentiation Formulas is:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f(t_{n+k}, y_{n+k}) \quad \beta_{k-1} = \ldots = \beta_0 = 0. \tag{20}$$

To obtain the BDF2 formula I will interpolate the function $y(t)$ using the points (t_n, y_n), (t_{n+1}, y_{n+1}),(t_{n+2}, y_{n+2}). The resulting polynomial $P(t)$ is

$$P(t) = y_n \frac{(t-t_{n+1})(t-t_{n+2})}{2h^2} + y_{n+1}\frac{(t-t_n)(t-t_{n+2})}{-h^2} + y_{n+2}\frac{(t-t_n)(t-t_{n+1})}{2h^2} \tag{21}$$

$P'(t_{n+2})$ is then used to approximate $y'(t_{n+2})$

$$P'(t) = \frac{1}{2h^2}y_n(t-t_{n+1}) - \frac{1}{h^2}y_{n+1}(t-t_n) + \frac{1}{2h^2}y_{n+2}[(t-t_n)+(t-t_{n+1})] + terms\ (t-t_{n+2}) \tag{22}$$

The terms with $t - t_{n+2}$ are omitted because they are null for $t = t_{n-2}$.

$$P'(t_{n+2}) = \frac{1}{2h^2}y_n(t_{n+2}-t_{n+1}) - \frac{1}{h^2}y_{n+1}(t_{n+2}-t_n) + \frac{1}{2h^2}y_{n+2}[(t_{n+2}-t_n)+(t_{n+2}-t_{n+1})] \tag{23}$$

$$P'(t_{n+2}) = \frac{1}{2h^2}y_n h - \frac{1}{h^2}y_{n+1}2h + \frac{1}{2h^2}y_{n+2}[2h+h] \tag{24}$$
$$P'(t_{n+2}) = \frac{1}{2h}y_n - \frac{2}{h}y_{n+1} + \frac{3}{2h}y_{n+2} \tag{25}$$
$$\tag{26}$$

The BDF2 formula can be derived from

$$P'(t_{n+2}) = f(t_{n+2}, y_{n+2}) \tag{27}$$
$$\frac{1}{2h}y_n - \frac{2}{h}y_{n+1} + \frac{3}{2h}y_{n+2} = f(t_{n+2}, y_{n+2}) \tag{28}$$
$$\frac{3}{2h}y_{n+2} - \frac{2}{h}y_{n+1} + \frac{1}{2h}y_n = f(t_{n+2}, y_{n+2}). \tag{29}$$

It can be finally expressed as

$$y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2}{3}hf(t_{n+2}, y_{n+2}). \tag{30}$$

Truncation Error

The general expression for the interpolation error is

$$E(t) = \frac{(t - t_{n+k}) \dots (t - t_n) f^{(k+1)}(\eta(t))}{(k+1)!}. \tag{31}$$

For the BDF2 formula, $k = 2$,

$$E(t) = \frac{(t - t_{n+2})(t - t_{n+1})(t - t_n) f^{(3)}(\eta(t))}{6} \tag{32}$$

The local truncation error is obtained using the maximum of the derivative of $|E'(t)|$

$$\begin{aligned}
\left|E'(t)\right| =& \frac{1}{6}\Big| -t_{n+2}(t - t_{n+1})(t - t_n) f^{(3)}(\eta(t)) && (33)\\
& -t_{n+1}(t - t_{n+2})(t - t_n) f^{(3)}(\eta(t)) && (34)\\
& -t_n(t - t_{n+2})(t - t_{n+1}) f^{(3)}(\eta(t)) && (35)\\
& -(t - t_{n+2})(t - t_{n+1})(t - t_n) f^{(4)}(\eta(t))\eta'(t)\Big| && (36)
\end{aligned}$$

This function $|E'(t)|$ has its maximum value for $t = t_n$ or $t = t_{n+1}$ or $t = t_{n+2}$. Choosing $t = t_n$ we simplify it to

$$\begin{aligned}
|E'(t_n)| &= \left| \frac{-t_n(t_n - t_{n+2})(t_n - t_{n+1}) f^{(3)}(\eta(t_n))}{6} \right| && (37)\\
&= \left| \frac{-t_n(-2h)(-h) f^{(3)}(\eta(t_n))}{6} \right| && (38)\\
&= \frac{h^2}{3} \left| t_n f^{(3)}(\eta(t_n)) \right| && (39)
\end{aligned}$$

The local truncation error can then be approximated, $T_n = \frac{h^2}{3} \left| t_n f^{(3)}(\eta(t_n)) \right| \approx O(h^2)$

Absolute stability

The characteristic polynomial for the BDF2 method is

$$t^2(3 - 2\bar{h}) - 4t + 1 = 0 \tag{40}$$

with roots

$$t_{12} = \frac{2 \pm \sqrt{1 + 2\bar{h}}}{3 - \bar{h}} \tag{41}$$

For $-\frac{1}{2} \leq \bar{h} < 0, t_{12}$ are both real, $t_1 \leq t_1(\bar{h} = -\frac{1}{2}) = 0.5$ and $t_2 < t_2(\bar{h} = 0) = 1$. Since the roots are less then 1 for the interval $-\frac{1}{2} \leq h < 0$, the method is A-stable in this interval.

For $\bar{h} < -\frac{1}{2}, t_{12}$ are complex and $t_{12} < t_{12}(\bar{h} = -\frac{1}{2}) = 0.5$. Since the roots are less then 1 for the interval $\bar{h} < -\frac{1}{2}$, the method is A-stable also in this interval.

I can hereby conclude that the method is A-stable for $\bar{h} \in (-\infty, 0)$.

Question 4

Stability for RK4

The 4-th order Runge-Kutta method, as showed before, is defined by

$$y_{n+1} = \left(1 + \frac{1}{6}hk_1 + \frac{1}{3}hk_2 + \frac{1}{3}hk_3 + \frac{1}{6}hk_4\right) y_n, \tag{42}$$

where

$$k1 = f(y_n) \tag{43}$$
$$k2 = f(y_n + \frac{h}{2}k_1) \tag{44}$$
$$k3 = f(y_n + \frac{h}{2}k_2) \tag{45}$$
$$k4 = f(y_n + hk_3). \tag{46}$$

Using $\bar{h} = h\lambda$, one can simplify this equation to:

$$y_{n+1} = \left(1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \frac{1}{6}\bar{h}^3 + \frac{1}{24}\bar{h}^4\right) y_n \tag{47}$$

The relation of the current iteration value y_n with the initial value y_0 is:

$$y_n = \left(1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \frac{1}{6}\bar{h}^3 + \frac{1}{24}\bar{h}^4\right)^n y_o \tag{48}$$

The absolute stability region satisfies the following inequality:

$$|1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \frac{1}{6}\bar{h}^3 + \frac{1}{24}\bar{h}^4| < 1 \tag{49}$$

Assuming h as a real number we have the following stability region:

$$-2.78529 < \bar{h} < 0 \tag{50}$$

This can be extended to a system of ODEs by using the largest modulus eigenvalue as λ found using `lambda = -eigs(A,1,'lm')` to be $\lambda = -7.8388262 \times 10^4$.

So,

$$h_{max} = \frac{-2.78529}{-7.8388262 \times 10^4} = 3.5531978 \times 10^{-5} \tag{51}$$
$$0 < h < 3.5531978 \times 10^{-5} \tag{52}$$

I tested various values of h around this value and found that the method produced `NaN` values for $h > 3.6 \times 10^{-5}$. The error for $h = h_{max}$ was $error(h_{max}) = 0.2231543$. The experimental h_{max} I found to be in the region: $3.55596 \times 10^{-5} < h_{max} < 3.55675 \times 10^{-5}$. This was noticeable because the error increased from 0.22624 to 5.1645.

Results

The following table shows the error and CPU elapsed time for each of the methods used and each of the different numbers of steps.

Method	**Number of steps**	**Error**	**CPU time (secs)**
ODE45	9445	1.155269×10^{-5}	$8.791882s$
CN	100	4.467899×10^{-3}	$208.038764s$
CN	1000	4.441078×10^{-4}	$514.197773s$
CN	10000	4.438412×10^{-5}	$3086.958397s$
BDF3	100	4.482679×10^{-3}	$188.455409s$
BDF3	1000	4.442484×10^{-4}	$557.765280s$
BDF3	10000	4.438552×10^{-5}	$3399.768021s$

Table 2: Final error and CPU elapsed time for each method and number of steps

For the Crank Nicolson method and the BDF3 formula the error is approximately $error \approx 4.4h$. The BDF3 formula produced slightly larger errors than the error of the Crank Nicolson method. These two methods are very comparable in speed and final error.

The Matlab method ODE45 has lower error than the others with 10000 steps. It is also noticeably faster, even when compared with the others using 100 steps. It is the clear choice to solve this ODE.

For 100 steps I would use the BDF3 formula, it has a slightly larger error but finished 20 seconds faster. For 1000 steps I would use the CN method, it has a slightly smaller error and finished 43 seconds faster. For 10000 steps the choice is also the CN method, it has a slightly smaller error and finished 313 seconds faster.

Question 5

The Lotka–Volterra equations describe the interaction between a population of predators and preys.

$$x'(t) = x(t)(\alpha - \beta y(t)) \tag{53}$$
$$y'(t) = -y(t)(\delta - \gamma x(t)) \tag{54}$$
$$x(0) = x_0 \tag{55}$$
$$y(0) = y_0. \tag{56}$$

The implementation on Matlab uses a 2D function $f(t) = [x'(t), y'(t)]$ and a vector $r = [x, y]$ to simplify the use of the 4-th order Runge Kutta method.

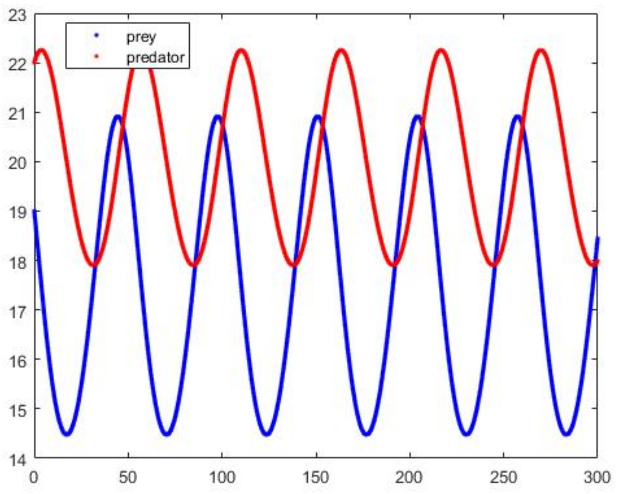

Figure 4: Evolution of the number of preys and predators.

Figure 4 shows the solution of these equations using the 4-th order Runge Kutta method explained before. The results are as expected. The increase of the number of preys sustains the increase in the number of predators, the number of preys then reduce and consequently so does the numbers of predators due to less food avaulability. This is the explanation for the sinusoidal evolution of these values with the predator having a lower amplitude and being delayed relative to the prey evolution.