Instructor: Dr. S. Hughes

Due date: Sun, March 17 (midnight)

Lab 7 - Ordinary Differential Equations

Keywords: Ordinary Differential Equations (ODEs), Coupled ODEs, RK4, Euler, Backward Euler, Applications and Examples: Harmonic Oscillator, Duffing Oscillator, Lorentz Attractor, Animations in Python.

Task: Go through all the questions and coding exercises below and hand in a single modularized Python code where each part runs each part of the question(s). Remember and include your surname at the beginning of the code name.

Code Submissions on onQ: Attach a single (preferable, or no more than several) .py (called LabX_YourLastName.py) and single .py.txt (same code, same name, just a different extension).

Marks: Coding Skills, Efficiency, Correct Results and Clear Graphs and Good Presentation of Results (6); Good use of Comments (2); Good Coding Structure and Readability (2). Total: (10)

Reminder: All codes must run under Python 3.9.x. and the Spyder IDE (check this before submitting). Section off the code by using #%% (and label section in code), and add useful comments, so we can step through it easily.

Acknowledgements: Please comment on any help that you received from your group members or others concerning this Lab assignment.

Some Background (in addition to lecture notes):

ODEs
Runge-Kutta
Lorenz system
Rubin Landau (video lecture ODEs 1)
Rubin Landau (video lecture ODEs 2)

Question 1

(a) Using the ODE *stepper* template on onQ, where you have an example of the Forward Euler ODE, which is called with the simple command:

```
y1 = odestepper('euler',fun, y0, ts),
```

add in a general Backward Euler ODE solver as well, so that you can call it in exactly the same way. Using $n = 10$ and then $n = 20$ points (uniform spacing) with a time array, spanning: $t = [0, 0.6]$, show the ODE solutions versus the exact solution for $\dot{y}(t) = -10t$, with $y(0) = 1$. The exact solution is $y(t) = e^{-10t}$. Plot the results, with a clear labelling of the different results. For reference, see Fig. 1 below.

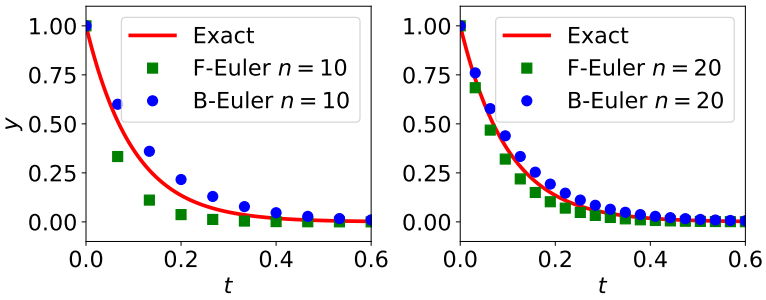

Figure 1: Numerical solutions of $y(t)$ versus exact solution.

(b) Now write a general (and vectorized) RK4 solver, which will also use the ODE stepper routine. Then implement a coupled ODE solution to the 1D Simple Harmonic Oscillator (SMO) problem:

$$\frac{d^2x(t)}{dt^2} = -\omega_0^2 x(t), \tag{1}$$

with $\omega_0 = \sqrt{k/m}$. Choose units: $m = k = \omega_0 = 1$, with the initial condition: $v(0) = 0$ and $x(0) = 1$, where $\dot{x} = v$ is the velocity in the x direction. Run your solution for 10 periods, where one period, in these units, is $T_0 = 2\pi$. Show the phase space solution as well as x, v versus time, for the two different step sizes of $dt = 0.01$ and $dt = 0.005$. Show the analytical solution with the latter graph, with key labels shown. For reference, see Fig. 2 for $dt = 0.01$.

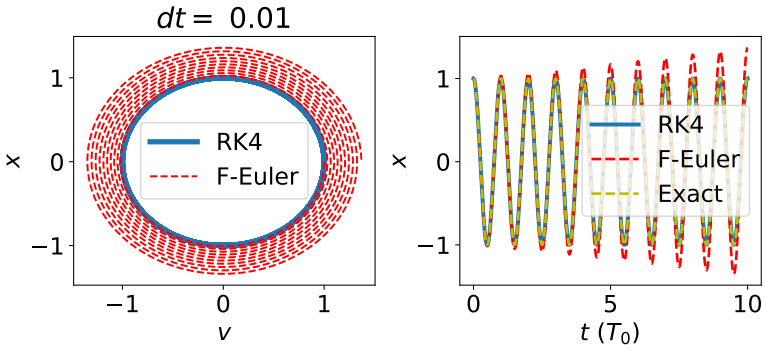

Figure 2: SHO ODE solutions in phase space, as well as versus the analytical answer (right graph).

Question 2

(a) Next, we will look at the Lorenz (not to be confused with Lorentz!) equations in the chaotic parameter regime known as the "Lorenz Attractor." The Lorenz equations are given by:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= rx - y - xz, \\ \frac{dz}{dt} &= xy - bz, \end{aligned} \tag{2}$$

originally derived by Edward Lorenz in 1963, for a model of fluid dynamics.

Choosing $\sigma = 10$, $b = 8/3$, and and $r = 28$, implement an RK4 solution for four periods ($T_0 = 2\pi$) using $dt = 0.01$. Show the 3D solution graphically for the two slightly different initial conditions, as shown the Fig. 3. Here I am using `ax = fig.add_subplot(projection='3d')` and `ax.view_init(azim=20, elev=29)`, but feel free to explore other options and routines. Also if you use %matplotlib auto (in the Spyder Console), you can rotate the graph.

From the Wikipedia article link: In popular media the "butterfly effect" stems from the real-world implications of the Lorenz attractor, i.e. that in any physical system, in the absence of perfect knowledge of the initial conditions (even the minuscule disturbance of the air due to a butterfly flapping its wings), our ability to predict its future course will always fail.

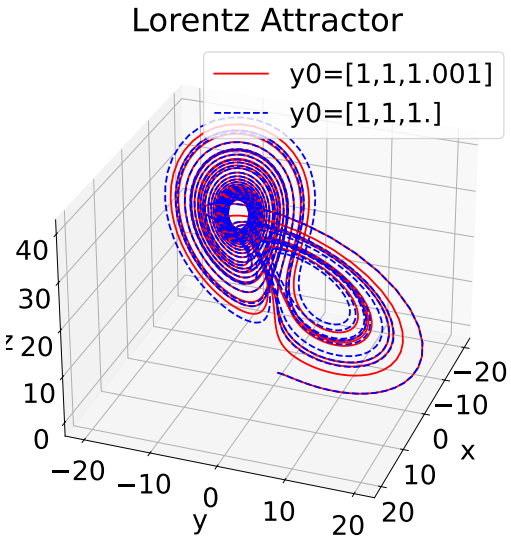

Figure 3: Lorentz Attractor - solution with two different different initial conditions.

(b) Wikipedia has a cool simulation of the Lorenz attractor, but we can do better - as we get closer towards mastering Python! Thus your final task for this lab is to show an animation of the Lorenz Attractor using the `animation.FuncAnimation` available `from matplotlib import animation`. An example mp4 played with such an animation is shown in the lecture notes (final page, and also on onQ). Use %matplotlib auto to show animations within the Spyder Console (or they wont show). There is no need to save an MP4 or animated Gif, though this is quite easy to do in Python.