Assignment #7

Math 1800: Mathematical Programming in Python

Instructions: Choose 3 of the following problems to work on. Submit your responses as Python text files, with the extension `.py`. Each file should include your name and the problem number. This problem set is due Thursday, March 02, by midnight.

- *Problem 7.0:* Consider a biological population whose size at time $t_0 = 0$ is $y_0 = 10$. If a population has a growth rate $r = 0.25$, an exponential growth model is

  $$y' = r\,y$$

  If there is a carrying capacity $k = 100$, a logistic growth model is

  $$y' = r\,y\,(1 - y/k)$$

  Use the Euler method to estimate the solutions of both population models over the interval $0 \leq t \leq 10$ and make a plot showing the first solution in red, and the second in blue.

- *Problem 7.1:* The exact solution of the logistic equation can be written as

  $$y(t) = \frac{k\,y_0\,e^{rt}}{k + y_0\,(e^{rt} - 1)}$$

  Supposing $t_0 = 0$, $y_0 = 10$, $r = 0.25$ and $k = 100$, use the Euler method with $n = 20$ to estimate the solution of $0 \leq t \leq 10$. In one plot, display your estimated solution in blue, and the exact solution in red.

- *Problem 7.2:* The Runge-Kutta method of order 2 is similar to the Euler method, but uses both `y[i-1]` and an intermediate value `ymid` in order to compute the next value `y[i]`. Assuming the step size is `dt`, the formula can be written as follows:

  $$\begin{aligned} tmid &= t_{i-1} + 1/2\,dt \\ ymid &= y_{i-1} + 1/2\,dt\,f(t_{i-1}, y_{i-1}) \\ y_i &= y_{i-1} + dt\,f(tmid, ymid) \end{aligned}$$

  Copy the file *euler_solve.py* into a new file *rk2_solve.py* and modify it so that it uses the Runge-Kutta method of order 2.

- *Problem 7.3:* Use the Euler method to solve the flame ODE over the interval $0 \leq t \leq 200$:

  $$\begin{aligned} y' &= y^2 - y^3 \\ t_0 &= 0 \\ y_0 &= 0.01 \end{aligned}$$

  Plot your solution; if it seems to jump up and down irregularly, try a larger value of n. You should expect a fairly smooth solution curve.

- *Problem 7.4:* Use `solve_ivp()` to solve the flame ODE over the interval $0 \leq t \leq 200$:

  $$\begin{aligned} t_0 &= 0 \\ y_0 &= 0.01 \\ y' &= y^2 - y^3 \end{aligned}$$

- *Problem 7.5:* Consider the (linear) pendulum ODE

$$u' = v$$
$$v' = -(g/l) * u$$

with parameter values

$$g = 9.81$$
$$l = 1$$
$$t_0 = 0$$
$$u_0 = \frac{\pi}{3}$$
$$v_0 = 0$$

This equation has a conserved quantity:

$$h(t) = g * l * u(t)^2 + v^2(t)$$

Use the Euler method to estimate the solution over the interval $0 \le t \le 20$, computing the value of $h(t)$ at each step. Plot $h(t)$ in red, and for scale, include the line $h(t) = 0$ in black. Does your estimated solution do a good job of conservation?

- *Problem 7.6:* Repeat Problem 7.5, but this time use `solve_ivp()`.

- *Problem 7.7:* Suppose we are given a second order ODE involving u''. (Recall that u'' is another way of writing the second derivative of u.) We have seen how to replace such a problem by two first order ODEs, defining a second variable $v = u'$.

  Consider the following third order ODE:

$$u''' - 4tu' - 2u = 0$$
$$u(0) = 1$$
$$u'(0) = 2$$
$$u''(0) = 3$$

  Convert this to a system of three first order ODEs by defining variables $v = u'$, $w = u''$. Create the Python function that defines the right hand side vector of this system, and a short Python script that would set up and solve this system over the interval $0 \le t \le 5$, using the `scipy` function `ivpsol()`. You do not have to actually run this program!

- *Problem 7.8:* The Lorenz equations are a famous simple model inspired by the problems of weather prediction. The variables can be represented as a three-component array y, and there are three parameters, β, ρ and σ, or in English, `beta, rho, sigma`. The equations have the form

$$y_0' = \sigma * (y_1 - y_0)$$
$$y_1' = y_0 (\rho - y_2) - y_1$$
$$y_2' = y_0 y_1 - \beta y_2$$

  Using the initial condition `[8,1,1]`, and the parameter values $\beta = 8/3, \rho = 28, \sigma = 10$, try to solve this system over the interval $0 \le t \le 40$. I want to see your Python script that defines the right side, and the commands that call `solve_ivp()`. If you do a time plot you may see that the three variables behave somewhat chaotically.

- *Problem 7.9:* The Arenstorf orbit was discussed in class on 24 February 2023 and in the notes *python_ode*. The system is presented as two second order differential equations. By adding variables xp and yp, you can rewrite the system as four first order equations

$$\frac{dx}{dt} = xp$$
$$\frac{dxp}{dt} = ...$$
$$\frac{dy}{dt} = yp$$
$$\frac{dyp}{dt} = ...$$

  Write the file *arenstorf_dydt(t,y)* that would evaluate the right hand sides of this system. Assume that M_e and M_m are supplied as global variables. Be careful not to get confused by the fact that the name y is being used in two ways here, first as the "vertical" coordinate of the satellite, but then also as the vector of length 4 holding the current solution value.

  You don't have to try to solve this system. I just want to see your right hand side function.