Exercise #12

31. March 2025

Exercises marked with a (J) should be handed in as or together with a Jupyter notebook.

Optional exercises will not be corrected

Problem 1. (Lipschitz continuity)

Determine whether the following functions are Lipschitz continuous for all $t, y \in \mathbb{R}$.

a) $f(t,y) = e^{-t^2}y$.

b) $f(t,y) = \dfrac{t^2 y}{1+t^2}$.

Problem 2. (4th order Runge-Kutta - (J))

The classical 4th order Runge–Kutta method is given as

$$
\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n) \\
\mathbf{k}_2 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1\right) \\
\mathbf{k}_3 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2\right) \\
\mathbf{k}_4 &= \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{k}_3) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4).
\end{aligned}
$$

a) Implement this method in Python.

b) Verify numerically that this method has convergence order $p = 4$.

You may use the example problem

$$y' = -2ty,$$
$$y(0) = 1.$$

Recall that the analytic solution of such problem is $y(t) = e^{-t^2}$.

Problem 3. (Numerical solution of ODEs - (J))

In this problem we will implement Euler's method, second order Taylor's method, and Heun's method, and use them to approximate the solution to the ODE,

$$y' = -ty + \sin(t), \quad y(0) = 2.$$

The exact solution to this equation is $y(t) = e^{-t^2/2}\left(2 + \int_0^t e^{s^2/2}\sin(s)\,ds\right)$. You can use the notebook `numerical-ode-3.ipynb` as a starting point.

a) Implement Euler's method, and compute an approximation of $y(1)$, using a step size equal to 0.1.

b) Do the same using Heun's method and the second order Taylor method. The second order Taylor method is given by

$$y_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2}f'(t_n, y_n),$$

where $f(t,y) = y'(t) = -ty + \sin(t)$ and thus $f'(t,y) := \frac{d}{dt}f(t,y) = \frac{d}{dt}(-ty + \sin(t))$.

c) We now want to approximate the convergence orders of these methods numerically. Recall that we defined the global error,

$$\varepsilon_g := \max_n |y(t_n) - y_n|.$$

If we assume that $\varepsilon_g(h) \approx Mh^p$, for some $M > 0$, we have,

$$\log\left(\frac{\varepsilon_g(h_1)}{\varepsilon_g(h_2)}\right) \approx p \log\left(\frac{h_1}{h_2}\right).$$

Compute the global error of the methods from a)-b) using $h_1 = 10^{-2}$ and $h_2 = 10^{-3}$, where $t_n = nh, n = 0, \dots, \frac{1}{h}$. Use this to approximate the convergence order, p, for each of the three methods.

d) We can also approximate the convergence order by plotting $\log(\varepsilon_g(h)) = \log(M) + p\log(h)$ versus $\log(h)$, and inspecting the slope of the function.

   Plot $\log(\varepsilon_g(h))$ versus $\log(h)$ for $h = 10^{-2}, 10^{-3}, 10^{-4}$ for each of the three methods.

Problem 4. (Runge–Kutta method - (J))

In this exercise we will study a Runge–Kutta method that is given by

$$
\begin{aligned}
k_1 &= f(t_n, y_n)\\
k_2 &= f\left(t_n + \frac{h}{3}, y_n + \frac{k_1}{3}\right)\\
k_3 &= f\left(t_n + \frac{2}{3}h, y_n - \frac{1}{3}k_1 + k_2\right)\\
k_4 &= f\left(t_n + h, y_n + k_1 - k_2 + k_3\right)\\
y_{n+1} &= y_n + \frac{h}{8}(k_1 + 3k_2 + 3k_3 + k_4)
\end{aligned}
$$

a) Present the method in the form of a Butcher tableau.

b) Decide the order of the method.

c) Implement this method in Python.

d) Verify the convergence order numerically. For this you can use the example problem

$$y' = 2ty, \qquad y(0) = 1,$$

   which has the analytical solution $y(t) = e^{t^2}$, on the interval $[0, 1]$.

The next exercises are optional and should not be handed in!

Problem 5. (System of ODEs)

Write the second order linear ODE,

$$\begin{aligned} 2y + y' + y'' + 1 &= 0, \\ y(0) &= 0, \\ y'(0) &= 2, \end{aligned}$$

as a linear system of first order ODEs, and perform one step of Euler's method with step size 1.

Solution.

We start by setting $w_1 = y$ and $w_2 = y'$. Inserted into the ODE, this gives

$$\begin{aligned} w_1' &= w_2 \\ w_2' &= -2w_1 - w_2 - 1. \end{aligned}$$

This can be expressed by

$$\boldsymbol{w}' = A\boldsymbol{w} + b,$$

where

$$\boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 \\ -2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

Also $\boldsymbol{w}(0) = (0, 2)$. One step of Euler's method with step size 1 in this case gives

$$\boldsymbol{w}_1 = \boldsymbol{w}(0) + A\boldsymbol{w}(0) + b = \begin{pmatrix} 0 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ -2 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}.$$

Problem 6. (Implementation of an ODE solver)

```python
import numpy as np

f = lambda t,y : 2/t**2*y
t0, tend = 1, 2
y0 = 1
N = 10

y = np.zeros(N+1)
t = np.zeros(N+1)
y[0] = y0
t[0] = a

for n in range(N):
    k1 = f(t[n],y[n])
    k2 = f(t[n]+0.5*h, y[n]+0.5*h*k1)
    y[n+1] = y[n] + h*k2

print('t=',t)
print('y=',y)
```

a) There are three bugs in this code. Two that prevent it from running at all, and one which causes a completely nonsense output. Find and correct the errors.

b) Which mathematical problem does this code intend to solve numerically?

c) Which specific algorithm has been applied to the problem? No specific name is required, but present the method in the form of a Butcher tableau, and decide the order of the method.

d) Find the first two elements of the NumPy vector `y`, given that point a) is accomplished.

Solution.

```
import numpy as np

f = lambda t,y : 2/ t**2*y
t0, tend = 1, 2
y0 = 1
N = 10

y = np.zeros(N +1)
t = np.zeros(N +1)
y[0] = y0
t[0] = t0                   #Assigning a starting time
h = (tend-t0)/N             #Need to define h

for n in range (N):
   k1 = f(t[n],y[n])
   k2 = f(t[n]+0.5* h , y[n]+0.5* h * k1)
   y[n+1] = y[n] + h*k2
   t[n+1] = t[n] + h        #Need to update timestep

print('t=',t)
print('y=',y)
```

a) The corrected version is written above, with comments for where the code is changed. The errors that made the code not run were that $t[0]$ was not set to t_0 but to some undefined variable a and that h was not defined. In addition, there were no computations of new timesteps, which made the output wrong.

b) This problem tries to solve the initial value problem

$$y' = \frac{2}{t^2}y, \qquad y(1) = 1,$$

on the interval $[1, 2]$.

c) The method presented as a Butcher tableau:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

This method is known as the explicit midpoint method. Next, we check the order conditions:

$p = 1$	$b_1 + b_2 = 0 + 1 = 1$	OK
$p = 2$	$b_1c_1 + b_2c_2 = 0 + 1 \cdot \frac{1}{2} = \frac{1}{2}$	OK
$p = 3$	$b_1c_1^2 + b_2c_2^2 = 0 + 1^2 \cdot \frac{1^2}{2^2} = \frac{1}{4} \neq \frac{1}{3}$	Not satisfied

We see that up to $p = 2$, the conditions are satisfied. The method is therefore of order 2.

d) If we run the code, we get that the first two elements of y are 1. and 1.19954649.