Exercise #7

02. October 2023

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

Problems marked with 4N or 4D must be submitted only by the respective course, while unmarked problems must be submitted by both courses.

Optional exercises will not be corrected.

A serious attempt in solving 70 – 75% of all the tasks must be done in **both** theory and coding (if present) in order to pass the exercise.

Problem 1. (4th order Runge-Kutta - **J**)

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

$$\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\
k_4 &= f(t_n + h, y_n + hk_3) \\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}$$

a) Implement this method in Python.

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

   You may use the example problem

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

   which is also used in the Jupyter notes. Recall that the analytic solution is $y(t) = e^{-t^2}$.

Problem 2.

Consider the initial value problem

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

a) Find the exact solution to the equation, then compute $y(0.4)$.

   Hint: You should obtain that $y(0.4) = 25/29$.

b) Perform 4 steps of Euler's method with $h = 0.1$. Compute the error at the last step, that is $e_4 := |y_4 - y(0.4)|$.

c) Perform 2 steps of Heun's method with $h = 0.2$. Compute the error at the last step, that is $e_2 := |y_2 - y(0.4)|$.

d) Perform 1 step of the classical 4th order Runge-Kutta method (as described in the problem avove) with $h = 0.4$. Compute the error $e_1 := |y_1 - y(0.4)|$. In each case, 4 function evaluations were needed. Which of the methods performed best?

Problem 3. (Numerical ODE-methods for a spring)

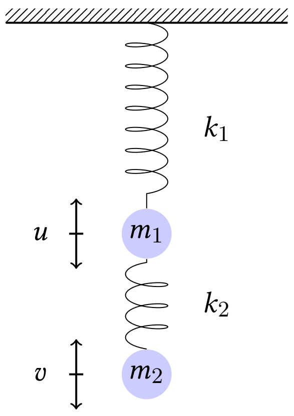
We consider the coupled mass-spring system

$$\begin{aligned} m_1 u'' &= -k_1 u + k_2(v - u), \\ m_2 v'' &= -k_2(v - u), \end{aligned}$$

with initial conditions

$$\begin{aligned} u(0) &= a, \qquad & u'(0) &= b, \\ v(0) &= c, \qquad & v'(0) &= d. \end{aligned}$$

Here u and v describe the (purely vertical) displacements of the masses m_1 and m_2 from the equilibrium position, and $k_1, k_2 > 0$ are the spring constants:



a) Rewrite this second order system as a system of four first-order ODEs.

b) Perform one step of Heun's method with step length $h = 0.1$ for the solution of this system. Use the following parameters:

$$\begin{array}{cccc} k_1 = 100, & k_2 = 200, & m_1 = 10, & m_2 = 5, \\ a = 0, & b = 1, & c = 0, & d = 1. \end{array}$$

c) Use both Euler's method and Heun's method in order to find a numerical approximation of the solution of this problem (with the parameters as above) on the interval $[0, 3]$. Test the step lengths 0.1, 0.01, and 0.001, and plot the results. (J)

d) From the principle of convervation of energy, it follows that the total energy in this mass-spring system remains constant for all times. For this system, this total energy is given as

$$E = \frac{m_1(u')^2}{2} + \frac{m_2(v')^2}{2} + \frac{k_1u^2}{2} + \frac{k_2(u-v)^2}{2}.$$

Test numerically, to which extent the energy E is conserved in the different numerical solutions which you have obtained in the previous step. (J)

Problem 4. (Implementation of an ODE solver)

```
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.