Dr Andreas Dedner

MA261

Assignment 1 (20% of total module mark) due Thursday 2nd Feburary 2022 at 12pm

Do not forget to provide us with the following information in a **legible** way and read through the regulations given below carefully! If you have any question ask me.
Note: we might be marking only some parts of each question.
Assignment is based on material from weeks 1, 2, and 3.

Student id	
Second part joint work with (enter student id of partner)	

Regulations:

- Python is the only computer language to be used in this course
- You are *not* allowed to use high level Python functions (if not explicitly mentioned), such as `diff, int, taylor, polyfit`, or ODE solvers or corresponding functions from `scipy` etc. You can of course use mathematical functions such as exp, ln, and sin and make use of `numpy.array`.
- In your report, for each question you should add the Python code, and worked-out results, including figures or tables where appropriate. Use either a Python script and a pdf for the report or combine everything in a Jupyter notebook. We need to be able to run your code.
- Output numbers in floating point notation (using for example the *npPrint* function from the quiz).
- A concisely written report, compiled with clear presentation and well structured coding, will gain marks.
- Do not put your name but put your student identity number on the report.
- The **first part** of each assignment needs to be done **individually** by each student.
- For the second part submission in pairs is allowed, in which case one student should submit the second part and both need to indicate at the top of the their pdf for the first part the student id of their partner. Both will then receive the same mark for the second part of the assignment.

Part 1

Q 1.1. [2/20]
Consider the ODE $y' = f(y)$ with initial condition $y(0) = y_0$ and the following approximation

$$y_{n+1} = y_n + \frac{h}{2}\Big(f\big(y_n\big) + f\big(y_n + hf(y_n)\big)\Big)$$

We use the notation from the lecture: h is the step size and $y_n \approx \underline{Y}(t_n)$ with $t_n = nh$.

(1) Consider the method applied to the linear ODE $y' = \lambda y$ with real valued $\lambda < 0$. For which values of λ is the method stable?
(2) Consider the method applied to the linear ODE $y' = \lambda y$ with imaginary $\lambda = i\alpha$. For which values of α is the method stable (i.e. the solution remains bounded)?

Q 1.2. [6/20]
Consider the ODE $y' = f(y)$ with initial condition $y(0) = y_0$ and the approximation from the first question

$$y_{n+1} = y_n + \frac{h}{2}\Big(f\big(y_n\big) + f\big(y_n + hf(y_n)\big)\Big)$$

We use the same notation as in the lecture where h is the step size and $y_n \approx \underline{Y}(t_n)$ with $t_n = nh$. We assume that f is Lipschitz and $\underline{Y} \in C^3$.

(1) Show that the truncation error satisfies $\tau_n = O(h^3)$.
Hint: make sure you also Taylor expand the term $f(y + hf(y))$. Also make good use of the ODE!
(2) Using $\tau_n = O(h^3)$ and Gromwall's lemma, prove that the method converges quadratically:

$$e_n = O(h^2) \qquad \text{(for } h \text{ small enough)} .$$

Hint: use that f is Lipschitz to derive something like

$$|e_{n+1}| \leq R(hL_f)|e_n| + \tau_n$$

with a suitable polynomial $R = R(\mu)$.

Q 1.3. [2/20]
Consider a scalar linear ODE $y' = \lambda y$ on $[0, T]$ with $\lambda \in \mathbb{R}$ fixed. Assume that an approximation is given by

$$y_{n+1} = R(\lambda h) y_n$$

where $R = R(\mu)$ is a non-constant polynomial.
Prove that such an approximation is never unconditionally stable.

Part 2

Hand in one program listing which covers all the questions given below in a single script/notebook reusing as much as possible. Avoid any code duplications (or explain why you decide to duplicate some part of the code).
Important: In your brief discussion of your result (a paragraph with a plot or a table is sufficient) refer back to the theoretical results discussed in the lecture or from assignments.

Q 2.0. [see online quiz on moodle]
Implement the following functions and test them individually. Take a look at the quiz questions which also include some tests and more detail on the function signature to use. After the quiz closes you can use the provided solutions if you encountered problems implementing the required functions.

(1) Write a function that computes a single step of the forward Euler method

$$y_{n+1} = y_n + hf(t_n, y_n)$$

given $h, t_n \in \mathbb{R}$ and $y_n \in \mathbb{R}^m$ and a general vector valued function f which should be provided as a function handle to your function.
(The solution to this is available in the warmup quiz).

(2) Write a function that computes a single step of the method from **Q 1.1**

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

given $h, t_n \in \mathbb{R}$ and $y_n \in \mathbb{R}^m$ and a general vector valued function f which should be provided as a function handle to your function.

(3) Write a function *evolve* to solve a vector valued ODE of the form

$$y'(t) = f(t, y(t)) \ , \qquad t \in (0, T) \ , \qquad y(t_0) = y_0 \ .$$

using an approximation of the form:

$$y_{n+1} = \Phi(t_n, y_n; h) \ .$$

The code should be general with respect to the dimension of the ODE and the right hand side f. The right hand side and method function Φ should both be parameters (function handles) to your function *evolve* together with t_0, y_0, T, N and use a fixed step size $h = \frac{T}{N}$. The method should return the vectors $(t_n)_{n=0}^N$ and $(y_n)_{n=0}^N$.

(4) Write a function *computeEocs* to compute the experimental order of convergence (EOC) for a given sequence $(h_i, E_i)_{i=0}^r$. So the function should return

$$\mathrm{eoc}_i = \frac{\log\left(\frac{E_i}{E_{i-1}}\right)}{\log\left(\frac{h_i}{h_{i-1}}\right)} \ , \qquad i = 1, \ldots, r$$

where $E_i = E_{h_i}$ is the error of a method with step size h_i.

Note: I explained the concept of *EOC* during the seminar in week 2 (recorder) and there is a short recorded lecture on this concept which will be uploaded during week 3. Basically eoc_i should be close to the convergence rate of the method tested (at least for h small enough).

Q 2.1. [2/20]
Test your implementatioqn using the 2x2 ODE: $y'(t) = f(t, y(t))$ and $y(0) = y_0$ with

$$f(t, y_1, y_2) = \begin{pmatrix} y_2 \\ y_2(\lambda - 2y_1) \end{pmatrix}$$

for $t \in [0, T]$ with $T = 10$.
With $\lambda = 1$ and $y_0 = (2, -2)^T$ this has the exact solution

$$\underline{Y}(t) = \begin{pmatrix} \frac{2e^t}{2e^t - 1} \\ \frac{-2e^t}{4e^{2t} - 4e^t + 1} \end{pmatrix}$$

Compute the maximum errors E_{h_i} and the EOCs for the sequence of time steps given by $h_i = \frac{T}{N_0 2^i}$ for $i = 0, \ldots, 9$ using $N_0 = 25$.

Q 2.2. [2/20]
Write a program to solve a vector valued ODE of the form

$$y'(t) = f(t, y(t)) \ , \qquad t \in (0, T) \ , \qquad y(0) = y_0 \ ,$$

using the method from **Q 1.1**

$$y_{n+1} = y_n + \frac{h}{2}\Big(f(t_n, y_n) + f\big(t_n + h, y_n + hf(t_n, y_n)\big)\Big) \ .$$

Use the same *evolve* method as before but with a different parameter Φ. Implementing this was part of the first quiz.
Make sure to optimized your implementation with respect to the number of calls to f.
Repeat the experiments from **Q 2.1**.

Q 2.3. [3/20]
Compare the results from **Q 2.1** and **Q 2.2** with respect to their efficiency. To check efficiency compare the errors of the two methods with respect to the number of f evaluations needed to compute the approximations $y_0, \ldots, y_N$. Make sure your implementation of the methods from **Q 2.2** does not make any unnecessary evaluations of f!
Note: you can but don't need to change your code to keep track of the evaluations of f needed. In this case it is sufficient to just explain the difference between the methods based on your implementation noting that for a given N, both methods call Φ N times. So, the only difference between the methods comes from the number of calls to f in your implementation of the different Φ functions.

Q 2.4. [3/20]
Investigate the approximations given by both methods on the time interval $[0, 3]$ for the ODE $y'(t) = f(t, y(t))$ using the *method of manufactured solutions* with the following exact solution

$$\underline{Y}(t) = \begin{cases} \sin(t) & t < \frac{\pi}{2} \\ e^{t - \frac{\pi}{2}} & t \geq \frac{\pi}{2} \end{cases}$$

We will chose a very simple version of the right hand side

$$f(t, y) = \begin{cases} \cos(t) & t < \frac{\pi}{2} \\ y & t \geq \frac{\pi}{2} \end{cases}$$

The results might not be so clear cut as for the previous problems. Make sure to give a clear explanation of what you observe and how it fits in with the theory, i.e., what we have proven in the lecture for the forward Euler method and what was shown in **Q 1.2**.