Dr Andreas Dedner

MA261

Assignment 2 (15% of total module mark) due Thursday 11th Feburary 2020 at 12pm

Do not forget to fill this out (or provide the same information in your submission) in a **legible** way and read through the regulations given below carefully! If you have any question ask me.
Assignment is based on material from weeks 2,3,4.

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

Mark:

Part 1			Part 2			Sum

Regulations:

- Matlab or Python are the only computer language to be used in this course
- You are *not* allowed to use high level Matlab or Python functions (if not explicitly mentioned), such as diff, int, taylor, polyfit, and 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 etc.
- Output numbers in floating point notation either using (for example in Matlab) *format longE* or *fprintf('%1.15e',x).* Have a look at the solution suggestion from the quizzes.
- Submit **one pdf file** with the answers to the math question (handwritten is fine) and with your report for the second part. This should include any tables, figures etc. you use to show your simulation results.
  A concisely written report, compiled with clear presentation and well structured coding, will gain extra marks.
- Submit **one code file** (either an m-code or python script) that can be run as is, i.e., without needing any other files. So copy and paste any functions you might have in other files in a main file. If you find that impossible to do, upload a zip file and explain why you could not provide a single file as requested.
- Do not put your name but put your student identity number on the report.
- Each assignment will be split into two parts, the **first part** 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 first page that they have worked together on this. Both will then receive the same mark for the second part of the assignment.

Part 1

Q 1.1. [3/30] Consider a scalar linear ODE $y' = \lambda y$ with $\lambda \in \mathbb{R}$ fixed. Assume that an approximation is given by

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

where $P(\mu) = \sum_{k=0}^{m} \alpha_k \mu^k$ is a non-constant polynomial.
Prove that $y_n \to 0$ for $n \to \infty$ can not hold for $\lambda < 0$ and arbitrary values of h.
Remark: this shows that such a method is never unconditionally stable.

Q 1.2. [6/30] Consider a variation of the enzymatic reaction discussed in the lecture. In the modified model n units of substrate S have to bind to one enzyme E to generate the compound SE. This is a reversible reaction with rates k_f, k_r. As in the original model the compound decomposes at a rate k into one unit of the product P and one enzyme. In the reaction network below C is used to denote the compound SE:

$$E + nS \underset{k_b}{\overset{k_f}{\rightleftharpoons}} C \xrightarrow{k} P + E$$

We assume that at the starting time there is no compound or product present in the system while S_0, E_0 are the initial values for the substrate and the enzyme.

Part a: Write down the stoichiometry matrix Γ and speed vector w.

Part b: Find two vectors c_1, c_2 that span the kernel of Γ^T and write down the resulting conserved quantities H_1, H_2.

Part c: Use the two conserved quantities to express the compound C and the enzyme E in terms of the product P, the substrate S, and the initial conditions for S, E.

Part d: Using the two conserved quantities and your equations from the previous parts, provide a system of ODE for S, P.

Q 1.3. [6/30]
Consider a difference equation

$$x_{k+1} = \Phi(x_k)$$

with a given smooth function $\Phi \in C^p$. Assume that for a given x_0 we have $x_k \to x$ for $k \to \infty$.
Show that if $\Phi(x) = x$ and $\Phi'(x) = \Phi''(x) = ... = \Phi^{(p-1)}(x) = 0$ then the order of convergence of $(x_k)_k$ to x is at least of order p.

Part 2

Hand in one program listing which covers all the questions given below in one single code reusing as much as possible. Avoid any code duplications (or explain why you decide to duplicate some part of the code). The problem considered is scalar but your code should be able to handle vector valued ODEs. So think of the scalar case as also being vector valued but with vectors of size one, i.e., not to have y being a float value. The implementation quiz will test a vector valued version.
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, e.g.:

- what have we discussed about the convergence rates of all the methods discussed below and does your result confirm the theory?
- if you are seeing convergence rates that do not match the theory then either you have a bug in the implementation or for some reason the theory is not applicable to this test case. You should comment on this in your report, i.e., explain why you're code is working correctly and the theroy is not applicable.

Note: it should now be possible to upload a code file. So your submission should consist of one pdf file and one code file which is self contained, i.e., can be executed without requiring any additional files.

Q 2.0. [see online quiz on moodle]
Implement the following two functions and test them individually (unit testing). 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 the root of a given function $F\colon \mathbb{R}^m \to \mathbb{R}^m$. The function F and its Jacobian should be passed to the function using function handles. In addition the function requires an initial guess x_0, a tolerance ε, and a maximum number of iterations K to perform before giving up. The iteration stops with the kth iterate x_k if either $|F(x_k)| < \varepsilon$ or $k = K$.
The function should return the final approximation x_k to the root and the number of steps k used. So if the returned number of steps equals the maximum number of iterations provided as parameters, i.e., $k = K$ then the Newton method failed to converge and any calling function should handle that case, e.g., terminate with an error.

(2) Write a function that computes a single step of the backward Euler method

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

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 together with its Jacobian as a function handles to your function. Use the Newton method to solve the nonlinear problem.

Q 2.1. [5/30]
Test your implementation of the Backward Euler method using the scalar ODE from the previous assignment:

$$y'(t) = (c - y(t))^2,\ y(0) = 1,\ c > 0$$

which has the exact solution

$$\underline{Y}(t) = \frac{1 + tc(c-1)}{1 + t(c-1)}\ .$$

Compute errors at the final time T and the EOC for the sequence of time steps given by $h_i = \frac{T}{N_i}$ with $N_i = N_0 2^i$ for $i = 0, \dots, 10$. Discuss the approximation using

$$c = 1.5\ , T = 10\ .$$

Q 2.2. [5/30]
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 following method (Crank-Nicholson)

$$y_{n+1} = \Phi(t_n, y_n; h) := y_n + \frac{h}{2}\big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\big) \ .$$

Use the same *evolve* method as before but replace the function handle for the forward Euler method by a function handle implementing the above function.
Repeat the experiments from **Q 2.1**.
Remember to refer back to the theory discussed in the lecture - also have a look at the example sheet from week 3.

Q 2.3. [5/30]
Investigate the approximations given by the two methods from the first assignment, i.e., the forward Euler method and the method from Q2.2 on the first assignment (Heun's method) for the following ODE $y'(t) = f(t, y(t))$ with $y(0) = 1$ using the following choice for f:

$$f(t, y) = \begin{cases} -y & t < \bar{t} \\ y & t \geq \bar{t} \end{cases} \ , \qquad \bar{t} = \frac{1}{\sqrt{2}} \ .$$

Solve up to $T = 1$ and as on assignment 1 compute experimental orders of convergence for both methods.
You will need to solve the ODE analytically to find $Y(T)$ so that you can compute the errors - simply find the solution Y_1 for $y' = -y$, $y(0) = 1$ up to $\bar{t}$ and then find the solution Y_2 to the initial value problem $y' = y$ with $y(\bar{t}) = Y_1(\bar{t})$. The solution Y of the original problem is then Y_1 for $t < \bar{t}$ and Y_2 for $t > \bar{t}$.
The results might not be so clear cut as before for this example. Make sure to give a clear explanation of what you observe and how it fits in with the theory.
Remark: for the example on the first assignment you should have seen quadratic convergence rates for Heun's method (Q1.2). If you haven't, recheck your code and compare with the code available on the moodle site.