Dr Andreas Dedner

MA261

Assignment 2 (20% of total module mark) due Thursday 23nd Feburary 2023 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 to 6.

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 following chemical reaction network

$$A + A \underset{k_r}{\overset{k_f}{\rightleftharpoons}} A^\star + A \ , \qquad A^\star \xrightarrow{l} P$$

where A is some molecule and P the product of the reaction. $A^\star$ is called a reaction intermediate. Assume given initial conditions $A_0, A_0^\star, P_0$.

(1) Derive the system of ODEs for $A, A^\star, P$ modelling the above reaction network. Using a conservative quantity derive a system of ODEs for only A, P.
(2) A typical model simplification approach is to make a *steady-state approximation* in which one assumes that the rate of production of a quantity equals the rate of consumption so that its time derivative is zero. In the above reaction it is often reasonable to make this assumption for $A^\star$.
Using the *steady-state approximation* that $A^\star$ does not change in time, derive a new system of ODEs involving only A, P.

Q 1.2. [3/20]

Consider a two stage explicit RK method given by the Butcher tableau $\begin{array}{c|cc} 0 & 0 & 0 \\ a & a & 0 \\ \hline & 1-\gamma & \gamma \end{array}$ where $a, \gamma \in [0,1]$.
We know that this method will always be first order and to make it second order we need to choose the constants so that $x := a\gamma = \frac{1}{2}$.
Instead of optimizing the order, we will now try to optimize the stability of the method. First consider the linear case $y' = \lambda y$ with real valued $\lambda < 0$. We are looking to maximize $h_0 = h_0(a, \gamma)$ so that the method is stable (i.e., $y_n \to 0$ for $n \to \infty$) for $h < \frac{h_0}{|\lambda|}$.

(1) Compute the stability bound h_0 and show that it only depends on the product $x = a\gamma$ of the two free parameters (as we have seen the order does).
(2) Find the value x_{opt} for x for which h_0 is largest so that $|y_n|$ remains bounded.
Explanation: if you take x slightly above x_{opt} and h slightly below h_0 then $|y_n| \to 0$ as originally required. Taking $x = x_{\text{opt}}$ is not quite enough - in a sense x_{opt} is an *infimum* and not a *minimum*. That makes the question slightly more difficult to formulate than I realized. That is why I am now only asking for boundedness of $|y_n|$.
If you want to you can make the following quite easy calculation: for any $\varepsilon > 0$ take $x = x_{\text{opt}}(1+\varepsilon)$ then the method will satisfy $y_n \to 0$ for $h < \frac{h_0}{1+\varepsilon}$. But as I said you don't need to show this.
(3) Consider the same two stage explicit Runge-Kutta method but now we take $\lambda = \alpha i$ with real $\alpha \neq 0$. For which choices of $x = a\gamma$ is there a positive $h_0 > 0$ so that for $h < \frac{h_0}{|\alpha|}$ the method is stable, i.e., satisfies $|y_n| \leq |y_0|$ for all n?

Extra: You don't have to write down the answers to the following follow on questions: the only one stage explicit Runge-Kutta method is the Forward Euler method. By adding a second explicit stage we have doubled the computation cost (roughly speaking). By chosing $x = x_{\text{opt}}$, how much larger can we take the time step compared with the Forward Euler method - is it worth doing? Is there any h_0 for the Forward Euler method for which the solution remain bounded in the case $\lambda = i\alpha$ - again do we gain by adding a step?

Q 1.3. [5/20]
Consider a Hamiltonian $H(x,p)$ and the induced Hamiltonian system

$$x' = \partial_p H(x,p) \ , \qquad p' = -\partial_x H(x,p) \ .$$

To simplify things we will assume that x,p are scalar.
For an approximation (x_n,p_n) define $H_n := H(x_n,p_n)$. We would like to find approximations with $H_n = H_0$, i.e., the method conserves the Hamiltonian in time. We will study three methods. For the first two we will only prove results in the case that the Hamiltonian is of the form

$$H(x,p) = \frac{1}{2}p^2 + \frac{\alpha}{2}x^2 \tag{*}$$

with some positive constant α. We will show that the third method exactly conserves a general Hamiltonian in each step.

- The first is the Forward Euler method.
- The second is the so called *semi-implicit* Euler method

$$x_{n+1} = x_n + h\partial_p H(x_n,p_n) \ , \qquad p_{n+1} = p_n - h\partial_x H(x_{n+1},p_n) \ . \tag{SI-E}$$

  As you can see this is a very simple modification of the standard Forward Euler method for the case of Hamiltonian systems. This is a type of *split method* - which in fact is *explicit* in spite of the name.
- The third method is fully implicit and also split:

$$\begin{aligned} x_{n+1} &= x_n + h\frac{H(x_n,p_{n+1}) - H(x_n,p_n)}{p_{n+1}-p_n} \ , \\ p_{n+1} &= p_n - h\frac{H(x_{n+1},p_{n+1}) - H(x_n,p_{n+1})}{x_{n+1}-x_n} \ . \end{aligned} \tag{FI-Method}$$

  Make sure you understand why this is a reaonable approximation of the Hamiltonian system...

(1) Consider the case of the quadratic Hamiltonian (*) and the approximation (x_n,p_n) to the Hamiltonian system from the Forward Euler method.
We have already seen simulations in the lecture that indicate that in this case the value of the Hamiltonian increases in each time step, i.e., $H_{n+1} > H_n$.
Prove that this is the case by showing that $H_n = (1+\alpha h^2)^n \, H_0$.

(2) Consider now the *semi-implicit* Euler method (SI-E). Show that when applying this method to the quadratic Hamiltonian (*) the method can be rewritten as

$$\begin{pmatrix} x_{n+1} \\ p_{n+1} \end{pmatrix} = A \begin{pmatrix} x_n \\ p_n \end{pmatrix} \ ,$$

with a suitable matrix A.
Note: In fact the method conserves a “perturbed Hamiltonian“ $E_h(x,p) := H(x,p) - h\frac{\alpha}{2}px$. One can use this to show that H_n stays $O(h)$ close to H_0 but we will not do that in the lecture.

(3) Consider now a general Hamiltonian and the approximation given by the *fully implicit* method (FI-Method). Making the assumption that $p_{n+1} \neq p_n$ and $x_{n+1} \neq x_n$, show that $H_n = H_0$ for all n.

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.
You do not need to include any testing you did of these method in the report.

(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 with respect to y. Use Newton's method to solve the nonlinear problem.

Q 2.1. [1/20]
Implement the *Crank-Nicholson* method given by
$$y_{n+1} = y_n + \frac{h}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\Big)\ .$$
We have seen in the seminar that this method converges with second order.
Repeat the first experiment from the first assignment comparing the forward Euler method, the backward Euler method, the method *Q11* from assignment 1, and the Crank-Nicholson method.
So you should compute the maximum errors and EOCs for all four methods for the ODE with right hand side
$$f(t, y_1, y_2) = \begin{pmatrix} y_2 \\ y_2(1 - 2y_1) \end{pmatrix}$$
for $t \in [0, 10]$ and $y_0 = (2, -2)^T$ using the time step sequence $h_i = \frac{T}{N_0 2^i}$ for $i = 0, \dots, 9$ using $N_0 = 25$.

Q 2.2 (you can discuss this together with Q2.3 and Q2.4 if you want). [2/20]
Consider the Hamiltonian system for $y = (x, p)$ induced by the Hamiltonian for a nonlinear mass-spring system $H(x, p) = \frac{1}{2}p^2 + V(x)$ where the potential is given by
$$V(x) := \frac{k^2 + 1}{2}x^2 - \frac{k^2}{2}x^r$$
with positive constants k, r.
For $r = 2$ the Hamiltonian system is a harmonic oscillator and since it's linear you can easily find the exact solution. For $r = 4$ the system is know as *Duffy equation* and the

exact solution for this problem is given by the *elliptic Jacobi functions* which are available from `scipy.special.ellipj`. For the initial conditions $y_0 = (0, 1)$ the solution is:

```
from scipy.special import ellipj
def Y(t):
    sn, cn, dn, _ = ellipj (t, k*k)
    return array ([ sn , cn * dn ])  # (x(t),p(t))
```

Do the following experiments for both $r = 2$ and $r = 4$. Use $k = 0.8$, $y_0 = (0, 1)$ as initial conditions, and simulate up to $T = 40$: compare errors and EOCs for the two second order methods (Q11 and Crank-Nicholson) and the backward Euler method using the time step sequence $h_i = \frac{T}{150}2^{-i}$ for $i = 0, \dots, 10$.
Add plots of the trajectories in phase space for $i = 0, 1$ for all three methods.

Q 2.3. [3/20]
Implement the *semi-implicit* Euler method discussed in Q1.3 above: You can restrict your implementation to scalar x, p and the case of a Hamiltonian of the form $H(x, p) = \frac{1}{2}p^2 + V(x)$. But of course you can also do the general case.
Repeat the experiments from Q1.2. How does the new method compare to the others? Which rate of convergence do you think the new method has?
In addition to the errors also discuss the relative error in the Hamiltonian for all methods, i.e., study the sequence (h_i, E_h^{Ham}) with $E_h^{\text{Ham}} := \max_n \frac{|H(y_n)-H(y_0)|}{|H(y_0)|}$.
Hint: You should be able to write a function mapping $y_n = (x_n, p_n)$ to $y_{n+1} = (x_{n+1}, p_{n+1})$ that you can use as stepper in your code. It is sufficient to implement this function for $y = (x, p)$ with scalar x, p but feel free to consider the general case. After doing that you should be able to use the code from Q2.2.

Q 2.4 (this question is perhaps a bit harder...) [4/20]
Implement the *fully implicit* method from Q1.3. Optimize the implementation for the case of a Hamiltonian of the form $H(x, p) = \frac{1}{2}p^2 + V(x)$. To optimize the implementation first rewrite the method in the form

$$x_{n+1} = x_n + h\varphi_x(x_n, p_n, x_{n+1}; h) \ , \qquad p_{n+1} = p_n + h\varphi_p(x_n, p_n, x_{n+1}) \ ,$$

using that $H(x, p) = \frac{1}{2}p^2 + V(x)$. Add your formulation to your report... It is okay to implement the method only for scalar x, p.
Repeat the experiments from Q1.3. How does the new method compare to the others? Which rate of convergence do you think the new method has?
Hint: After you reformulated the method to make use of the special form of the Hamiltonian, you will need to use Newton's to compute x_{n+1} from given (x_n, p_n). After that step, p_{n+1} is explicitly computable.
Depending on how you ended up defining φ_x, you might have to put some thought into finding a good initial guess to use for Newton's method. Explain what you did and why you think that is a good choice for the initial guess. If you couldn't get it to work, give a short description of what you tried - it might still earn you some marks.