Dr Andreas Dedner

MA261

Assignment 3 (20% of total module mark) due Friday 10th March 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. **[3/20]** Consider the following model for a fish population

$$B'(t) = rB(t)\left(1 - \frac{B(t)}{K}\right) - \beta\frac{B(t)^2}{\alpha^2 + B(t)^2}$$

where r, K, β, α are given constants. The model assumes growth according to the logistic equation where K is the carrying capacity and the second term describes a loss rate due to predators. It approaches a constant loss rate β for a large fish population ($B \to \infty$). For a small fish population the loss is quadratic in the size of the population.
We assume that the dimensions for B and t are `fish` and `week`, respectively. So $[B] =$ `fish` and $[t] =$ `week`.

(1) What dimensions do the constants r, K, β, α have to have for the model satisfies dimensional homogeneity?
(2) Define non-dimensional $\tau = t/T$ and $x(\tau) = B(T\tau)/F$ and derive the model

$$x' = x(1-x) - \Pi_1\frac{x^2}{\Pi_2 + x^2}$$

by chosing suitable characteristic time and fish scales T, F. Check that the new parameters Π_1, Π_2 are dimensionless.
(3) Using a different choice for T, F derive the model

$$x' = \Pi_3 x(1 - \Pi_4 x) - \frac{x^2}{1+x^2}\ ,$$

with dimensionless (check) parameters Π_3, Π_4.

Q 1.2. **[2/20]** Consider the homogeneous ODE $y' = f(y)$ with $\partial_y f(y)$ invertible for all y. The *stiff Taylor scheme* is an explicit and surprisingly enough *L-stable* method of the form

$$(*) \qquad y_{n+1} = y_n + \Big(\exp(h\partial_y f(y_n)) - I_m\Big)\big(\partial_y f(y_n)\big)^{-1} f(y_n)\ .$$

Here we use the matrix exponential $\exp(A)$ and $I_m \in \mathbb{R}^{m,m}$ is the identity matrix.
Remark: we will look at a Runge-Kutta type version of this idea in the second part.

(1) For a given t_n we can rewrite our ODE in the form

$$y'(t) = F(y(t)) := D_n y(t) + g_n(y(t))\ , y(0) = \underline{Y}(t_n)$$

with $D_n := \partial_y f(\underline{Y}(t_n))$ and $g_n(y) := f(y) - D_n y$. Note that $F(y) = f(y)$ so this problem still has the same exact solution $\underline{Y}$ which can be written in the form

$$\underline{Y}(t) = \exp(D_n(t-t_n))\underline{Y}(t_n) + \int_{t_n}^{t} \exp(D_n(t-\tau))g_n(\underline{Y}(\tau))\,d\tau\ .$$

Derive the method (*) by using the approximation $g_n(\underline{Y}(\tau)) \approx g_n(\underline{Y}(t_n))$ and as usual replacing $\underline{Y}(t_n), \underline{Y}(t_{n+1})$ by y_n, y_{n+1}, respectively.
(2) Show that the method is exact for any linear system of ODE $y' = Ay$ (so the method is L-stable).

Q 1.3. **[5/20]** Consider an ODE $y'(t) = f(y(t))$. An invariant (or *first integral*) of this ODE is a function $H\colon \mathbb{R}^m \to \mathbb{R}$ that satisfies $\nabla H(y) \cdot f(y) = 0$ for all $y \in \mathbb{R}^m$. From the definition it follows that if $\underline{Y}$ is the solution to the ODE then $\frac{d}{dt}H(\underline{Y}(t)) = 0$ using chain rule and therefore $H(\underline{Y}(t)) = H(\underline{Y}(0))$ for all t (compare with the discussion of conserved quantities and invariances for kinetic equations and the Hamiltonian systems).
Consider in the following a s-stage Runge-Kutta method with consistency order of $p \geq 1$ given by a Butcher tableau consisting of α, β, γ.

(1) Show that the Runge-Kutta method conserves any linear invariant, i.e., for invariants of the form $H(y) = c \cdot y$ with some fixed $c \in \mathbb{R}^m$ show that for all n: $H(y_n) = H(y_0)$.

(2) Prove that under the condition

$$\gamma_i \beta_{i,j} + \gamma_j \beta_{j,i} = \gamma_i \gamma_j$$

for $i, j = 1, \ldots, s$ the Runge-Kutta method maintains quadratic invariances of the form $H(y) = y \cdot Cy$ with a matrix $C \in \mathbb{R}^{m,m}$.

Hint:

Consider $H(z) = z \cdot Cz$ which implies $\nabla H(z) = (C + C^T)z$ so that H is an invariant if $(C + C^T)z \cdot f(z) = z \cdot (C + C^T)f(z) = 0$ for any vector z.

Use that for any vector a, b we have $(a+b) \cdot C(a+b) = a \cdot Ca + a \cdot (C + C^T)b + b \cdot Cb$ to show that the remainder R in $H(y_{n+1}) = H(y_n) + R$ is

$$R = h \sum_i \gamma_i y_n \cdot (C + C^T)k_i + h^2 \sum_{ij} \gamma_i \gamma_j k_j \cdot Ck_i \ .$$

Now you need to prove $R = 0$ under the given condition on β, γ which follows for example from using that $Y_{n,i} \cdot (C + C^T)k_i = Y_{n,i} \cdot (C + C^T)f(Y_{n,i}) = 0$ with $Y_{n,i} = y_n + h \sum_{j=1}^s \beta_{ij} k_j$ which allows us to replace $y_n \cdot (C + C^T)k_i$ in the expression for R.

(3) Use the above to find the values for d so that the Runge-Kutta method given by the Butcher tableau $\begin{array}{c|cc} d & d & 0 \\ 1-d & 1-2d & d \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$ satisfies the conditions from part (2) and therefore maintains quadratic invariances of the form $H(y) = y \cdot Cy$.

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 a function to performs one step of a general diagonally implicit Runge-Kutta method. The two vectors and matrix defining the DIRK should be passed in to the function and the function needs to work for vector valued ODEs.
Once you have implemented a function of the form `def y = dirk( f,Df, t0,y0, h, alpha,beta,gamma)` you can define a *stepper* fixing `alpha,beta,gamma` which you can then use in your `evolve` method, e.g., given the vectors and matrix `alphaCN,betaCN,gammaCN` defining the Crank-Nicholson method (provided in the lecture handout) the stepper is `CrankNicholson = lambda f,Df,t0,y0: dirk(f,Df,t0,y0,alphaCN,betaCN,gammaCN)`
The rest of your code should then work without change. You can test your code by comparing the results you get with your old steppers (`forwardEuler`, `backwardEuler`,`Q11`,`crankNicholson`). The errors should be (close to) identical.

Q 2.1. [2/20]
Test your implementation of the Diagonally Implicit Runge-Kutta methods using our test ODE from the previous assignments: $y'(t) = F(y(t))$ and $y(0) = y_0$ with

$$f(t, y_1, y_2) = \begin{pmatrix} y_2 \\ y_2(1 - 2y_1) \end{pmatrix}$$

for $t \in [0, 10]$ and $y_0 = (2, -2)^T$.
Compute the maximum errors and EOCs using the time step sequence $h_i = \frac{T}{N_0 2^i}$ for $i = 0, \ldots, 9$ using $N_0 = 25$ for the following three DIRK methods:

$$\begin{array}{c|cc} d & d & 0 \\ 1-d & 1-2d & d \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \text{with } d = 0, \tfrac{1}{4}, \tfrac{1}{2} + \tfrac{\sqrt{3}}{6}.$$

First check the order conditions for all three method to determin if the methods have consistency order equal to 1, equal to 2, or ≥ 3. Then discuss if the numerical results match your expectations.

Q 2.2. [3/20]
Note: Q2.3 is an extension of Q2.2 and you should combine both into one report if you can - Q2.3 includes the experiments done here so feel free to skip this question. I just added this question for anyone who wants to only do the easier part of Q2.3.

Consider the pendulum modelled by the Hamiltonian $H(x, p) = \frac{1}{2}p^2 - \omega_0^2 \cos(x)$. The problem has the exact solution:

```
from scipy.special import ellipj, ellipk
k = np.sin(x0/2)   # x0: initial condition
K = ellipk(k*k)
def Y(t):
    sn, cn, dn, _ = ellipj(K-omega0*t, k*k)
    x = 2*np.arcsin(k*sn)
    p = 2/sqrt(1-(k*sn)**2) * cn*dn * k*(-omega0)
    return array([x,p])
```

We set $\omega_0 = 4$ and simulate on the interval $[0, 100]$.
Compute errors and EOCs for the two implicit two stage DIRK methods from the previous question using the sequence $N_i = 500\ 2^i$ for the number of time steps with $i = 0, \ldots, 7$. In addition to the usual error vs. h plots and discussing the EOCs also discuss how well each method conserves the Hamiltonian at least for some time steps sizes, i.e., for $i = 1$ and $i = 3$.
Carry out experiments with initial conditions $(x_0, p_0) = (0.05\pi, 0)$ and $(x_0, p_0) = (0.6\pi, 0)$.

Note: some of the methods might not be stable (either $|y_n| \to \infty$ or failure of the Newton method to converge) for all values N_i and your code should be able to handle that (and you should remark on the cases where this occurred in your report).

Important: like always, don't forget to discuss how the results you observed relate to the theory. What do we know about the orders of the three methods and also have a look at your results from Q1.3 of this assignment.

Q 2.3. [5/20]
The following extends Q2.2 but is a bit more involved, requiring the implementation of a new method without the help of the quizzes:

Introduction:

Consider an ODE of the form $y' = f(t, y) = My + F(t, y)$ where the assumption is that the main dynamics is given by the linear part My while the non-linearity $F(t, y) = f(t, y) - My$ is only a small perturbation. We can obtain an ODE for $z = \exp(-Mt)y$ (using the matrix exponential):

$$z' = -M\exp(-Mt)y + \exp(-Mt)y' = \exp(-Mt)F\big(t, \exp(Mt)z\big) =: G(t, z)$$

For this ODE we can compute an approximation z_n to the exact solution $\underline{Y}(t_n)$ using for example a Runge-Kutta method. Then an approximation to the original solution $\underline{Y}(t_n)$ is given by $y_n := \exp(Mt_n)z_n$.
Note that if $F = 0$, i.e., $f(t, y) = My$ then $G(t, z) = 0$ and $z_n = z_0 = y_0$ for all n and therefore $y_n = \exp(Mt_n)y_0$ which is the exact solution to $y' = My$.
Scheme of that type are called *exponentially fit* method or *exponential* integrators. The main property being that they solve linear problems exactly. They are often used for Hamiltonian systems with a Hamiltonian close to linear.

Example:

Consider the Forward-Euler method for $z' = G(t, z)$, i.e., $z_{n+1} = z_n + hG(t_n, z_n)$. Multiplying with $e^{Mt_{n+1}}$:

$$y_{n+1} = e^{Mt_{n+1}}\Big(z_n + h\exp(-Mt_n)F\big(t_n, \exp(Mt_n)z_n\big)\Big)$$

Note that $e^{Mt_{n+1}} = e^{Mt_n}e^{Mh}$ so

$$y_{n+1} = e^{Mh}\Big(y_n + hF\big(t_n, y_n\big)\Big)\ .$$

While the Backward-Euler method $z_{n+1} = z_n + hG(t_{n+1}, z_{n+1}$ for $z' = G(t, z)$ leads to the following method:

$$y_{n+1} = e^{Mt_{n+1}}z_{n+1} = e^{Mh}e^{Mt_n}z_n + he^{Mt_{n+1}}e^{-Mt_{n+1}}F\big(t_{n+1}, \exp(Mt_{n+1})z_{n+1}\big) = e^{Mh}z_n + hF\big(t_{n+1}, y_{n+1}\big)$$

Important to note that, in both cases, we do not need to carry out the expensive computation of the matrix exponential e^{Mt_n} in each time step but only need e^{Mh} which for fixed h we can compute in advance.

Method:
We will implement an *exponentially fit* DIRK. These are based on a standard DIRK method with a Butcher tableau given by α, β, γ. The exponentially fit version is then

$$k_i = e^{-M\alpha_i h} F\Big(t_n + \alpha_i h, e^{M\alpha_i h}\big(y_n + h\sum_{j=1}^{i} \beta_{ij}\ k_j\big)\Big)\ , \quad y_{n+1} = e^{Mh}\Big(y_n + h\sum_{i=1}^{s} \gamma_i k_i\Big)\ .$$

Again note that we can pre-compute the required matrix exponentials for a given step size, i.e., $(e^{M\alpha_i h})_{i=1}^s$, $(\big(e^{M\alpha_i h}\big)^{-1})_{i=1}^s$, and e^{Mh}.

Example:
Let's look again at the Forward-Euler method with $s = 1$, $\alpha_1 = 0, \gamma_1 = 1$ and $\beta_{11} = 0$:

$$k_1 = F\big(t_n, y_n\big)\ , \qquad y_{n+1} = e^{Mh}\Big(y_n + hk_1\Big) = e^{Mh}\Big(y_n + hF\big(t_n, y_n\big)\Big)\ .$$

While the Backward-Euler method ($s = 1$, $\alpha_1 = 1, \gamma_1 = 1, \beta_{11} = 1$) leads to

$$k_1 = e^{-Mh} F\Big(t_n + h, e^{Mh}\big(y_n + hk_1\big)\Big)\ , \qquad y_{n+1} = e^{Mh}\Big(y_n + hk_1\Big)$$

Task:
Implement a new function
`def y = eDirk( F,DF, t0,y0, h, alpha,beta,gamma, exphMs, exphMsInv, exphM)`
to solve a general ODE of the form $y' = My + F(t, y)$. The first two additional parameters are the tuples $(e^{\alpha_i hM})_{i=1}^s$ and $(e^{-\alpha_i hM})_{i=1}^s = (\big(e^{-\alpha_i hM}\big)^{-1})_{i=1}^s$, respectively; the final parameter is e^{hM}. Since the matrix exponentials are expensive to compute, the required matrices from the matrix exponential should be computed only once in advanced before calling `evolve`.
Note: that the standard `numpy.exp` function does not work for matrices, to compute e^A one needs to use `from scipy.linalg import expm`.
Note: the first two argument are only the nonlinear part F of the right hand side and the Jacobian of F wrt. y so the actual right hand side is $f(t, y) = My + F(y)$ and $Df = M + DF$. We do not need M directly only the given matrix exponentials. So can setup the stepper using for example something like

```
stepper = lambda f,Df,t0,y0,h: (
              edirk(lambda t,y:f(t,y)-M.dot(y),
                    lambda t,y:DF(t,y)-M, t0,y0,h,
                    alpha,beta,gamma,
                    exphMs,exphMsInv,exphM)
          )
```

Experiment:
We extend the experiment from the previous question comparing the two implicit 2 stage DIRK methods with their exponentially fit counterparts. Write the Hamiltonian flow in the form

$$y' = f(t, y) := My + F(t, y)$$

for $y = (x, p)$ and with the matrix

$$M = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix}$$

Compare all four methods by repeating the experiments from Q2.2 using the two different initial conditions. Focus as in Q2.2 on the errors, eocs, and conservation of H.