Dr Andreas Dedner

MA261

Assignment 3 (15% of total module mark) due Thursday 4th March 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 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 $H(\underline{Y}(t)) = H(\underline{Y}(0))$ if $\underline{Y}$ is a solution to the ODE (compare with similar results for the Hamiltonian systems and conserved quantities for kinetic equations).

Prove that any Runge-Kutta method maintains 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 $H(y_n) = H(y_0)$.

Q 1.2. **[5/30]** Consider a Hamiltonian system with a separable Hamiltonian $H\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ of the form $H(x,p) = T(p) + V(x)$ where $T, V \in C^2$ have strictly positive second derivatives T'', V'' - as is the case with the mass-spring system.

Show that when applying the forward Euler method to solve the Hamiltonian system the value of the Hamiltonian will increase in each time step (recall example from first week).
Hint: Taylor expand $H(x_{n+1}, p_{n+1})$.

Q 1.3. **[7/30]** Consider the ODE $y'(t) = f(t, y(t))$ with exact solution $\underline{Y} \in C^3$ satisfying $\underline{Y}'(t) \neq 0$ and $\underline{Y}''(t) \neq 0$ for all $t \in [0, T]$.

Consider an *explicit method* of the form

$$y_{n+1} = y_n + a_n \exp(b_n h) + c_n.$$

Find a_n, b_n, c_n which may depend on y_n, f and its derivatives at (t_n, y_n) such that the method has consistency order 2 and is L-stable.
Hint: in the formula for the truncation error, use Taylor expansion of $\underline{Y}$ and expand $\exp(b_n h)$ around 0. Then eliminate as many terms with low powers of h by choosing suitable a_n, b_n, c_n as you can. Use this choice to construct an explicit method of the form given above with the desired properties.

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 theory is not applicable.

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 `function y = rungeKutta( f,Df, t0,y0, h, alpha,beta,gamma)` you can define a *stepper* which fixes `alpha,beta,gamma` which you can then use in your `evolve` method, e.g.,
`heun = @(f,Df,t0,y0) rungeKutta(f,Df,t0,y0,alphaHeun,betaHeun,gammaHeun)`
where for example `alphaHeun,betaHeun,gammaHeun` are the vectors and matrix defining the Heun method. The rest of your code should then work without change. You should test your code by comparing the results you get with your old steppers (`forwardEuler`, `backwardEuler`,`heun`,`crankNicholson`). The errors should be (close to) the same.

Q 2.1. [**5/30**]
Test your implementation of the Runge-Kutta method using our test ODE from the previous assignments:

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

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 $i = 0, \ldots, 10$ and $N_0 = 20$ using Heun, Crank-Nicholson, and the third order DIRK method given in the lecture:

$\frac{1}{3}$	$\frac{1}{3}$	0
1	1	0
	$\frac{3}{4}$	$\frac{1}{4}$

by using the new method for all three - so not your old implementation of Heun or CN.
Discuss the approximation using

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

Add a plot showing all three methods in one graph with h on the x-axis and the error on the y-axis (have a look at the plots in the solution given for assignment 1 uploaded to the moodle site). Disuss what you can observe from this plot.

Q 2.2. [**5/30**]
Solve the Hamiltonian system for the Hamiltonian $H(x,p) = \frac{1}{2}p^2 + \frac{1}{2}x^2$ using Crank-Nicholson and the midpoint methods given by

$\frac{1}{2}$	$\frac{1}{2}$
	1

Plot the time evolution of the Hamiltonian $(t_n, H(x_n, p_n))$ for both methods and discuss your observations. Use a large enough T (e.g. $T = 100$) and test at least two values for h.

Q 2.3. [**5/30**]
Consider the linear ODE with $f(y) = \lambda y$ with $\lambda = q(-1 + i)$ with real $q > 0$. Determine the stability bounds for the Forward-Euler and Heun method, i.e., for each method find an upper bound for $h_0(q)$ in terms of q so that the method is stable for $h < h_0$.
Rewrite the ODE as a system for $y_1 = \Re y, y_2 = \Im y$ and use the two methods to solve this system, assuming your above computation resulted in a positive h_0. Show that the stability bounds are sharp by solving the problem using $q = 1$ and $q = 10$ each with $h = 0.95h_0(q)$ and $h = 1.05h_0(q)$. So perform four computations for each method for which you got $0 < h_0(q) < \infty$.