Dr Andreas Dedner

MA261

Assignment 4 (15% of total module mark) due Thursday 18th 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. **[6/30]** Consider the LLM

$$y_{n+2} + ay_{n+1} + by_n = hcf(y_{n+i})$$

with fixed constants a, b, c. Answer the following questions for each $i \in \{0, 1, 2\}$:

(i) Give conditions for a, b, c so that the method is sure to converge.
(ii) Find values for a, b, c for the methods to have consistency order 2 (if possible).

Q 1.2. **[9/30]** Consider the non-linear oscillator given by

$$x''(t) + x(t) + \varepsilon\big(x^2(t) - 1\big)x'(t) = 0\ , \qquad x(0) = x_0, x'(0) = 0$$

which is a non-dimensionalized form of the van der Pol equation. It can be shown that this problem has a solution that will (eventually) be periodic (the model has a so called limit cycle). Since the problem is homogeneous we can start anywhere on the cycle, i.e., choose a point with $x' = 0$ (which should explain the initial conditions).
For the following you will need to be able to solve linear ODEs of the form $y'' + y = F(t)$ where the right hand side is of the form $F(t) = F_0 \cos(\omega t)$ (or $F(t) = F_0 \sin(\omega t)$) for some ω. The general form of the solution is with constants of integration C_1, C_2:

$$Y(t) := \begin{cases} \frac{1}{1-\omega^2} F(t) + C_1 \cos(t) + C_2 \sin(t) & \omega \neq 1\ , \\ \frac{1}{2} t\tilde{F}(t) + C_1 \cos(t) + C_2 \sin(t) & \omega = 1\ . \end{cases}$$

where $\tilde{F}' = F$. This should be known from MA133 or e.g. follow this link: Example 6.1.4 and Example 6.1.5.
Note that the problem is linear so a right hand side of the form $A\cos(\omega_1 t) + B\sin(\omega_2 t)$ can be handled with the same formula, i.e., the solution is the sum of the solutions Y_1, Y_2 given above with right hand sides $F(t) = A\cos(\omega_1 t)$ and $F(t) = B\sin(\omega_2 t)$, respectively. The central thing to take into account is the difference in the solution for $\omega \neq 1$ and $\omega = 1$ - the second case is know as *secular solution*.

Part A [4/30]

(i) Using regular perturbation theory of the form $x_0(t) + \varepsilon x_1(t) + O(\varepsilon^2)$ derive a system of ODEs for x_0, x_1.
(ii) Solve the ODE you obtain for x_0, x_1 under the assumption that we are looking only for periodic solutions.

Part B [5/30]

Using a expansion up to $O(\varepsilon^3)$ and after substituting x_0, x_1 into the ODE for x_2 one obtains

$$x_2'' + x_2 = \frac{1}{4}\cos(t) + 2C\sin(t) - \frac{3}{2}\cos(3t) + 3C\sin(3t) + \frac{5}{4}\cos(5t)$$

with some free parameter C.
You can see that the right hand side contains two terms with $\cos(t)$ and $\sin(t)$ which lead to secular terms in x_2. There is only one free constant C to choose which is not enough to eliminate both of these terms. So we can not obtain a periodic expansion above $O(\varepsilon^2)$. To construct a periodic expansion of higher order we need to allow the frequency of the oscillator to adapt to ε. To this end we define a *stretched time variable* $\tau = \omega t$ where the frequency ω also has an expansion of the form $\omega = 1 + \epsilon\omega_1 + O(\varepsilon^2)$.

(i) Derive an ODE for the function $y(\tau) = x(\frac{\tau}{\omega})$.
(ii) Use the expansion for ω and $y_0(t) + \varepsilon y_1(t) + O(\varepsilon^2)$ to derive a system of ODEs for y_0, y_1 (which should depend on ω_1).
(iii) Find periodic function y_0, y_1 - what choice for ω_1 does this require?

Remark: Expanding both ω and y up to $O(\varepsilon^3)$ leads to an equation for y_2 that looks so unpleasant that I didn't want to force you to write it down. After subsituting y_0, y_1, ω_1 into the ODE for y_2 one obtains

$$y_2'' + y_2 = (4\omega_2 + \frac{1}{4})\cos(\tau) + 2C\sin(\tau) - \frac{3}{2}\cos(3\tau) + 3C\sin(3\tau) + \frac{5}{4}\cos(5\tau)$$

with some constant C. It is now possible to obtain a periodic solution y_2 by choosing C, ω_2 in such way that the secular terms are removed.

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). This time the graphs and the discussion of the simulation results are of particular interest.
Have a look at the code provided on the moodle page. The aim is to simulate the *van der Pol* oscillator discussed in part 1 (and in the second part of the lecture in the context of Hamiltonians). We will use two RK methods (one fully explicit one DIRK) which I've also added. So you can solve the problem using either one of two fixed time step RK methods. I've also added code so that the problem can be solved using two build-in RK methods (one explicit, one implicit), which will provide a reference.
Start by having a look at the four possible simulations. Note especially that the build-in methods do not use a constant time step but that it adapts to the solution. Think about ways to visualize this by for example overlaying the logarithm of the time step with the actual solution on one graph or coloring the phase space portrait according to the time step used - I would be interested to see what that looks like; so if you tried that and decided it is not so informative mention that in the report. A lot of this assignment is about coming up with ways to visualize the outcome of simulations and comparing different methods, which is a major challenge in computational mathematics.

Q 2.1. [5/30]
Change the given problem by choosing $\mu = 100$ and $T = 200$, which makes the problem much stiffer. We don't know the exact solution but it is possible to obtain an estimate for the period of the oscillation, which for these parameters is around 162.64. Find suitable values for N (which will be different for the explicit and the DIRK method) so that the fixed time step method produce solutions with a relative error in the period of $1e-4$. My suggestion is to automate this by starting with a fairly large value for N and then increasing N by a factor of 2 until the desired accuracy for the period is reached (you'll need to come up with a way of computing the period given a sequence (t_n, y_n) - you could try to find an index K so that y_K is close to y_0, i.e., we have moved around the cycle at least once. Since T is only slightly larger then the expected period there should only be one such value with K close to N. The approximate period is then t_K.
Compare the fixed time step simulations with the simulations using the build-in methods, where you now need to find a suitable value for the tolerance so that the relative error in the period is below $1e-4$. You can experimentally find such a period as explained above, starting with some large tolerance and reducing that by a factor (e.g. of 2 or 10) until you reach the desired accuracy.
Discuss how the build-in methods compare with the fixed time step ones with respect to number of steps required and also how the explicit methods compare to the implicit ones. You can also estimate the computational time used if you want to - there are build-in functions for doing that called timeit or add a counter to the function f to find out the number of evaluations of the right hand side required (you can use a `global` variable which you increment each time f is called).
Remark: if you have problems with finding the period for the approximate solution just choose a tolerance for the adaptive methods and then try to find values for N so that the

graphs of the fixed time step simulations look similar. Just mention this in the report and I'll give partial marks for that.

Q 2.2. **[5/30]**
Now implement your own adaptive time stepping method - the idea is explained in the recording for week 9. I've provided a template for the implementation. Using the test cases from above compare your implementation with the build-in versions again by trying to obtain a relative error in the period below $1e-4$. Note that the build-in implementation is of course not identical to the one I'm suggestion so you might need different tolerances. A short summary of the method is given on the final page of the assignment.

Q 2.3. **[5/30]** Bringing it all together: we start with the reaction network with four reactions

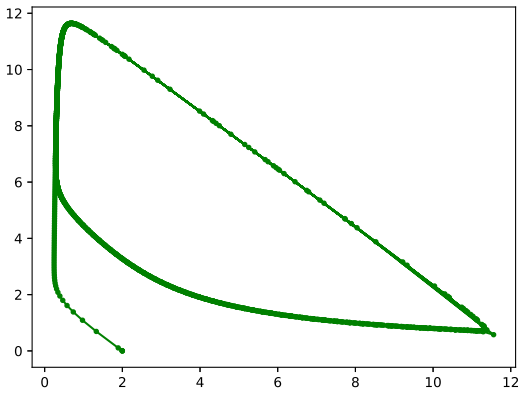
$$A \xrightarrow{k_1} X\ , \quad B + X \xrightarrow{k_2} Y + D\ , \quad 2X + Y \xrightarrow{k_3} 3X\ , \quad X \xrightarrow{k_4} E$$

(i) Derive the system of ODEs for the species X, Y assuming that the species A, B, D, E are constant, i.e., we assume that there are further reactions that make them conserved quantities. So just don't write down ODEs for A, B, D, E!

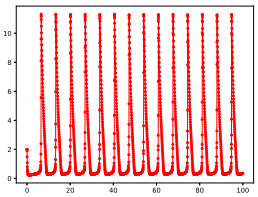
(ii) Assuming time is measured in seconds and that the species A, B, D, E, X, Y have units *mole per l*, e.g., $[A]$ =mole/l, which units do the reaction rates k_i have to have so that the system of ODEs satisfy dimensional homogeneity - or is the model not dimensional homogeneous?

(iii) Simulate the problem with your explicit adaptive time stepping method, just choose a suitable (i.e. good looking) value for the tolerance. That is, the solution is supposed to look good not the tolerance... Use initial conditions $(2, 0)$, $T = 100$, $k_1 = k_2 = k_3 = k_4 = 1$, $A = 2$, and $B = 8$. If you didn't get the adaptive methods to work use the build-in ones for some partial marks.

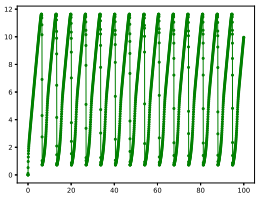
The interesting aspect of this system is that it has a periodic solutions (limit cycle) for any initial conditions.



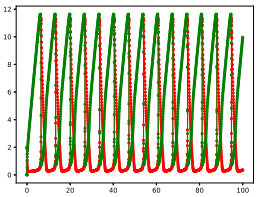
Phase space plot



X over time



Y over time



X and Y over time

Adaptive time stepping based on embedded RK methods:
The idea of automatically computing a time step is based on so called pairs of *embedded Runge-Kutta methods* where the same stages are used to compute two different approximations y_{n+1}, y^*_{n+1} by using two different vectors γ, γ^*:

$$\begin{array}{c|cccc} \alpha_1 & 0 & & & 0 \\ \alpha_2 & \beta_{2,1} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ \alpha_m & \beta_{m,1} & \dots & \beta_{m,m-1} & 0 \\ \hline & \gamma_1 & & \gamma_{m-1} & \gamma_m \\ & \gamma^*_1 & & \gamma^*_{m-1} & \gamma^*_m \end{array}$$

So there is little extra computational cost to compute botg y_{n+1}, y^*_{n+1} compared to simply computing y^*_{n+1}. We assume that the order of consistency of the method using γ is p while the consistency using γ^* is $p+1$. One can now estimate the error done in this step by computing $E := |y_{n+1} - y^*_{n+1}$ and use this to estimate a good value for the time step to take. Details on where that formula comes from can be found in the recording.
Algorithm: Given t_n, y_n and a initial step size h_n, e.g., the one used for the previous time step:

```
h = h_n
E = 2 * TOL
while E > TOL
  compute y_{n+1} and y*_{n+1} using the embedded RK methods
  E = |y_{n+1} - y*_{n+1}|
  if E < 1e-15
    need to come up with some strategy, e.g., use h_max
  end
  H = 0.9h (TOL/E)^{1/p}  % p is order of higher order RK method
  if H < 1e-10
    error: scheme is not working
  end
  h = H
end
h = min{h_max, h}  % want to avoid a too large time step

t_{n+1} = t_n + h
h_{n+1} = h  (initial time step for next iteration)
```

where $h_{\max}$ and TOL are fixed constant chooses a-priori.