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	u1902391
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.

Q 1.1

$y_{n+1} = y_n(1 + P(\lambda h))$ is a difference equation with the solution:

$$y_n = y_0(1 + P(\lambda h))$$

P is a non-constant polynomial so it is of order at least 1. This means, $\forall\lambda$, as $h \to \infty$, $|P(\lambda h)| \to \infty$. Hence, we can always find some value of h such that $|(1 + P(\lambda h)| > 1$ and the difference equation does not converge. Therefore, $y_n \to 0$ is not guarenteed for arbitrary h.

Q 1.2

The three equations being used are:

$$\begin{aligned}
R1 :&E + nS & \to^{k_f} C \\
R2 :&C & \to^{k_b} E + nS \\
R3 :&C & \to^{k} P + E
\end{aligned}$$

a) If the columns form left to right are $R1, R2, R3$ and rows from top to bottom are E, S, C, P:

$$\Gamma = \begin{pmatrix} -1 & 1 & 1 \\ -n & n & 0 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \qquad w = \begin{pmatrix} k_f ES \\ k_b C \\ kC \end{pmatrix}$$

b)

$$\Gamma^T = \begin{pmatrix} -1 & -n & 1 & 0 \\ 1 & n & -1 & 0 \\ 1 & 0 & -1 & 1 \end{pmatrix}$$

Which can kernel spanned by:

$$c_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad c_2 = \begin{pmatrix} 0 \\ 1 \\ n \\ n \end{pmatrix}$$

Which gives the following conserved quantities:

$$\begin{aligned}
H_1 &= E + C \\
H_2 &= C + nS + nP
\end{aligned}$$

c) As H_1 and H_2 are conserved:

$$\begin{aligned}
E + C &= E_0 + C_0 \\
&= E_0 \\
C + nS + nP &= C_0 + nS_0 + nP_0 \\
&= nS_0
\end{aligned}$$

The bottom equation gives:

$$C = n(S_0 - S) - nP$$

Rearranging the top equation and substituting:

$$\begin{aligned}
E &= E_0 - C \\
&= E_0 - n(S_0 - S) + nP
\end{aligned}$$

d) From the kinematic equations, we have:

$$\begin{aligned}\frac{dS}{dt} &= -nk_f ES^n + nCK_b \\ \frac{dP}{dt} &= kC\end{aligned}$$

From the conserved quantities, we know $C = n(S_0) - nP$ and $E = E_0 - n(S_0 - S) + nP$. Substituting these in:

$$\begin{aligned}\frac{dS}{dt} &= -nk_f(E_0 - n(S_0 - S) + nP)S^n + n(n(S_0 - S) - nP)k_b \\ &= -nk_f S^n(E_0 - n(S_0 - S) + nP) + n^2 k_b(S_0 - S - P) \\ \frac{dP}{dt} &= k(n(S_0 - S) - nP) \\ &= kn(S_0 - S - P)\end{aligned}$$

Q 1.3

As $\phi \in C^p$ we can write a Taylor expansion of $\phi(x)$ for x in some interval. As we are given x_k converges to some $x^\star$ all x_k must be contained in some bounded closed interval $[a, b]$ such that $x^\star, x_0 \in [a, b]$.

$$\begin{aligned}\phi(x_k) &= \phi(x^* + (x_k - x^\star)) \\ &= \phi(x_k^\star) + (x_k - x^\star)\phi'(x^\star) + ... + \frac{1}{(p-1)!}(x_k - x^\star)^{p-1}\phi^{(p-1)}(x^\star) + \frac{1}{p!}(x_k - x^\star)^p \phi^{(p)}(\eta)\end{aligned}$$

Where $\eta \in [x^\star, x_0]$ As $\phi(x^\star) = x^\star$ and $\phi^{(k)}(x^\star) = 0$ for $1 \leq k \leq p - 1$:

$$\begin{aligned}\phi(x) &= x^\star + \frac{1}{p!}(x_k - x^\star)^p \phi^{(p)}(\eta) \\ |\phi(x) - x^\star| &= |\frac{1}{p!}(x_k - x^\star)^p \phi^{(p)}(\eta)|\end{aligned}$$

As ϕ is smooth in C^p, $|\phi^{(p)}|$ must be bounded on the closed interval $[a, b]$ by some positive constant M. Hence, if for a given x_0 $x_k \to x^\star$:

$$\begin{aligned}|\phi(x_k) - x^\star| &= |\frac{1}{p!}(x - x^\star)^p \phi^{(p)}(\eta)| \\ |x_{k+1} - x^\star| &= \frac{M}{p!}|x_k - x^\star| \\ \frac{|x_{k+1} - x^\star|}{|x_k - x^\star|^p} &= \lambda \in \mathbb{R}\end{aligned}$$

Therefore, x_k converges with order p.

Q2.1

```
          i                      h_i                |Y(t_n)-y_N|               Eocs
--------------------    --------------------    --------------------    --------------------

0.00000000000000e+00    5.00000000000000e-01    6.09010973175295e-03    0.00000000000000e+00
1.00000000000000e+00    2.50000000000000e-01    3.07519247609056e-03    9.85791511418352e-01
2.00000000000000e+00    1.25000000000000e-01    1.54605614205727e-03    9.92084002552072e-01
3.00000000000000e+00    6.25000000000000e-02    7.75292344943557e-04    9.95780383642428e-01
4.00000000000000e+00    3.12500000000000e-02    3.88233669742721e-04    9.97815178291284e-01
5.00000000000000e+00    1.56250000000000e-02    1.94266588619652e-04    9.98887444789591e-01
6.00000000000000e+00    7.81250000000000e-03    9.71711062527358e-05    9.99438498530621e-01
7.00000000000000e+00    3.90625000000000e-03    4.85950536843749e-05    9.99717918864353e-01
8.00000000000000e+00    1.95312500000000e-03    2.42999079822503e-05    9.99858623856359e-01
9.00000000000000e+00    9.76562500000000e-04    1.21505500298991e-05    9.99929227625674e-01
1.00000000000000e+01    4.88281250000000e-04    6.07542411446005e-06    9.99964593786392e-01
```

EOCs suggest convergence of order 1 as predicted by theory of Backwards Euler method. The convergence appears to be significantly better early on than the forward Euler method which matches the diagram we saw at the start of the module.

Q2.2

```
          i                      h_i                |Y(t_n)-y_N|               Eocs
--------------------    --------------------    --------------------    --------------------

0.00000000000000e+00    5.00000000000000e-01    3.65016178470956e-04    0.00000000000000e+00
1.00000000000000e+00    2.50000000000000e-01    9.06264951496816e-05    2.00995561261687e+00
2.00000000000000e+00    1.25000000000000e-01    2.26183063865015e-05    2.00244198443914e+00
3.00000000000000e+00    6.25000000000000e-02    5.65219573789300e-06    2.00060757447579e+00
4.00000000000000e+00    3.12500000000000e-02    1.41290034783026e-06    2.00015171200313e+00
5.00000000000000e+00    1.56250000000000e-02    3.53215804160811e-07    2.00003791469325e+00
6.00000000000000e+00    7.81250000000000e-03    8.83033715037840e-08    2.00000946839922e+00
7.00000000000000e+00    3.90625000000000e-03    2.20758005209376e-08    2.00000276797672e+00
8.00000000000000e+00    1.95312500000000e-03    5.51895085187937e-09    1.99999981135660e+00
9.00000000000000e+00    9.76562500000000e-04    1.37973765745869e-09    2.00000005804412e+00
1.00000000000000e+01    4.88281250000000e-04    3.44930750628691e-10    2.00001532372991e+00
```

Eocs imply convergence of order 2 as suggsted by theory for the Crank Nicholson method.

Q2.3

To compute the exact solution, first solve in $0 \leq y < \frac{1}{\sqrt{2}}$. As $y' = -y$ get $y_1(t) = e^{-t}$ For $y \geq \frac{1}{\sqrt{2}}$, get solution $y_2(t) = e^{t+c}$ for some $c \in \mathbb{R}$. The Forward Euler method assumes $f \in C$ so we want:

$$\lim_{y \to \frac{1}{\sqrt{2}}^-} y_1(t) = \lim_{y \to \frac{1}{\sqrt{2}}^+} y_2(t)$$

Hence need $e^{-\frac{1}{\sqrt{2}}} = e^{\frac{1}{\sqrt{2}}+c}$ which implies $c = -\frac{2}{\sqrt{2}}$ Hence, overall, the exact solution is:

$$Y(t) = \begin{cases} e^{-t} & 0 \leq t < \frac{1}{\sqrt{2}} \\ e^{t-\frac{2}{\sqrt{2}}} & t \geq \frac{1}{\sqrt{2}} \end{cases}$$

Forward Euler Method

```
          i                      h_i                |Y(t_n)-y_N|                  Eocs
--------------------    --------------------    --------------------    --------------------

0.00000000000000e+00    5.00000000000000e-02    6.95697462859909e-02    0.00000000000000e+00
1.00000000000000e+00    2.50000000000000e-02    3.12139089278222e-02    1.15627102161483e+00
2.00000000000000e+00    1.25000000000000e-02    1.11780348051145e-02    1.48152246548459e+00
3.00000000000000e+00    6.25000000000000e-03    9.13350437514981e-03    2.91426162389441e-01
4.00000000000000e+00    3.12500000000000e-03    4.01917701273313e-03    1.18426838672569e+00
5.00000000000000e+00    1.56250000000000e-03    1.44769526965305e-03    1.47314216147474e+00
6.00000000000000e+00    7.81250000000000e-04    1.18989420590909e-03    2.82924648050901e-01
7.00000000000000e+00    3.90625000000000e-04    5.45328682772017e-04    1.12563536418306e+00
8.00000000000000e+00    1.95312500000000e-04    2.22820311700400e-04    1.29124528832684e+00
9.00000000000000e+00    9.76562500000000e-05    6.15096828173733e-05    1.85699530924269e+00
1.00000000000000e+01    4.88281250000000e-05    4.53760622824984e-05    4.38882118604458e-01
```

Heun Method

```
          i                      h_i                |Y(t_n)-y_N|                  Eocs
--------------------    --------------------    --------------------    ---------------------

0.00000000000000e+00    5.00000000000000e-02    2.38997236555359e-02     0.00000000000000e+00
1.00000000000000e+00    2.50000000000000e-02    7.26413474275867e-03     1.71813106830088e+00
2.00000000000000e+00    1.25000000000000e-02    1.08894772493584e-03     2.73785626452029e+00
3.00000000000000e+00    6.25000000000000e-03    3.00219508714150e-03    -1.46308302952193e+00
4.00000000000000e+00    3.12500000000000e-03    9.34877802564626e-04     1.68316802004354e+00
5.00000000000000e+00    1.56250000000000e-03    9.91434298611704e-05     3.23718872860205e+00
6.00000000000000e+00    7.81250000000000e-04    4.16508113014968e-04    -2.07075552287859e+00
7.00000000000000e+00    3.90625000000000e-04    1.58341786852145e-04     1.39530256052834e+00
8.00000000000000e+00    1.95312500000000e-04    2.92533463230882e-05     2.43636846633330e+00
9.00000000000000e+00    9.76562500000000e-05    3.52921861273003e-05    -2.70747133062389e-01
1.00000000000000e+01    4.88281250000000e-05    3.02474280955423e-06     3.54446441670762e+00
```

Both methods accurately predict the answer, which makes sense as the right hand side is always defined. Alsol geometrically, as Euler's method follows gradient lines, which are almost always defined, we would expect Euler's method to produce a correct approximation. However, piecewise nature of f means $Y \in C^0$ ie not differentiable so the truncation term may not be bounded. Hence Eocs don't converge. This is also seen in figure 1, as the error does not reduce monotonically, leading to points where the error may even increase, thus affecting the EOC calculation. That said, the fact the error does shrink as is clear on the figure, seems to explain why the absolute value of the EOCs seems to remain bounded between 1.5 and 3.5. This may be due to Y still being Lipshitz on the closed interval $[0, 1]$ preventing extreme absolute changes in in y from step to step.

Furthermore, the error of Euler's method bounds that of Heun's method, which fits the theory that Heun's method is more accurate, being of a higher order when Y is smoother. So whilst neither method is able to reduce errors as fast as usual, they do seem to converge, with Heun being the quickest.

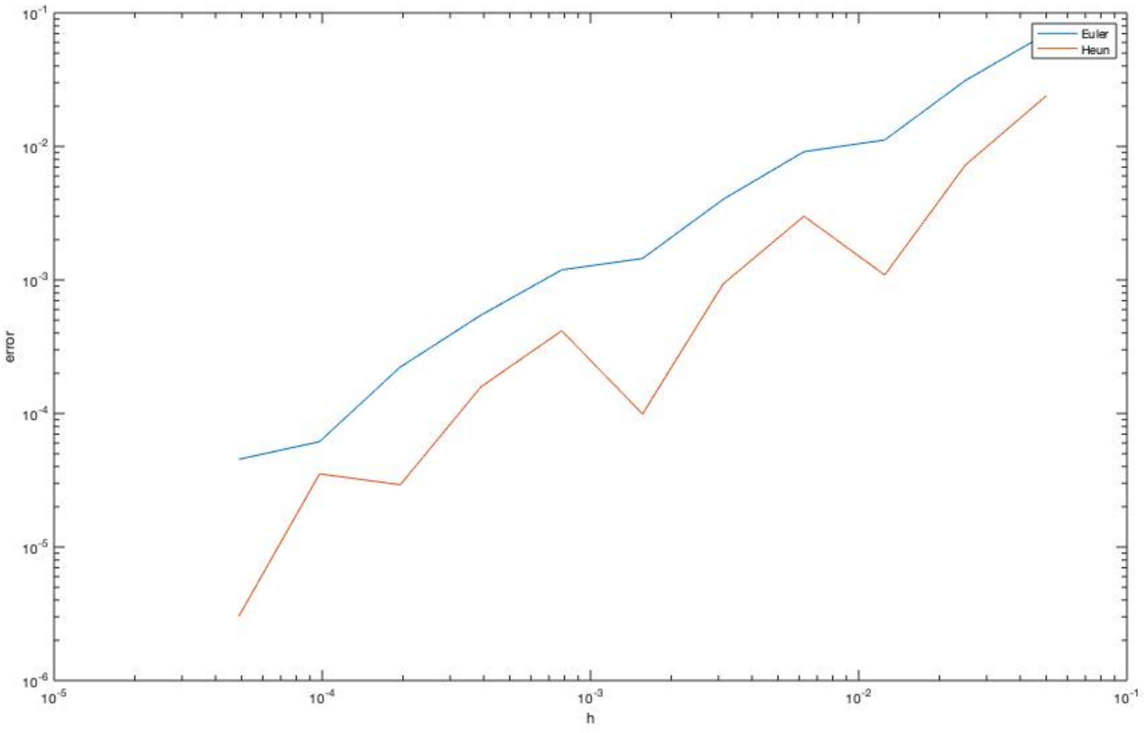

Figure 1: Plot of Error against step size for Euler and Heun Methods