Exercises: Initial value problems
Numerical Analysis, FMNF10, 2018

1. Consider the initial value problem

\[ y'(t) = -y(t)^2 + \cos(t), \quad t > 0, \]

with the initial value \( y(0) = 2 \).

a) Approximate \( y(0.1) \) by using the explicit Euler method with the step size \( \Delta t = 0.05 \).

b) Explain what is meant by the statement: “The explicit Euler method has a convergence order of \( p = 1 \).”

c) Your friend has implemented a different method and solved the problem for various values of \( \Delta t \). The resulting global errors \(|y(0.1) - u_n|\) are given in the graph below, where \( u_n \) denotes the approximation of \( y(0.1) \). Estimate the convergence order \( p \) of your friend’s method.

2. Rewrite the second order differential equation

\[ v''(t) = v(t)^2 + v'(t), \quad v(0) = 1, \quad v'(0) = 0, \]

as a system of first order equations (standard form). Do not forget the initial value.

3. Consider the MATLAB program

```matlab
lambda = -1000;
N = 10;
u(1) = 1;
dt = 1/N;
for n = 1:N
    u(n+1) = u(n) + dt*(lambda*u(n));
end
```
a) Which numerical method is implemented in the code, and which problem does it solve?

b) If we would run the code it would give a very disappointing result. Explain why this is the case.

c) Modify the code (without changing the problem being solved) so that the numerical approximation is improved.

4. The stability of a time stepping method can be checked by applying it to the test equation

\[ y'(t) = \lambda y(t), \quad y(0) = \alpha. \]

When this problem is approximated one obtains a recursion

\[ u^{n+1} = R(\lambda \Delta t)u^n, \]

where \( R \) is a polynomial if the method is explicit and a rational function, i.e., the quotient of two polynomials, if the method is implicit. The stability region is then defined as the set

\[ S = \{ \lambda \Delta t \in \mathbb{C} : |R(\lambda \Delta t)| \leq 1 \}. \]

Find and plot the stability region in the complex plane for the trapezoidal rule, where

\[ u^{n+1} = u^n + \frac{1}{2} \Delta t \left( f(t_n, u^n) + f(t_{n+1}, u^{n+1}) \right). \]

5. The classical Runge–Kutta schemes is given by

\[
\begin{align*}
  k_1 &= f(t_n, u_n), \\
  k_2 &= f(t_n + \frac{1}{2} \Delta t, u_n + \frac{1}{2} \Delta t k_1), \\
  k_3 &= f(t_n + \frac{1}{2} \Delta t, u_n + \frac{1}{2} \Delta t k_2), \\
  k_4 &= f(t_n + \Delta t, u_n + \Delta t k_3), \\
  u^{n+1} &= u^n + \frac{1}{6} \Delta t (k_1 + 2k_2 + 2k_3 + k_4). 
\end{align*}
\]

a) Write a Matlab function

\[
\text{function un_plus_1 = RK(dt,tn,un)}
\]

that evaluates one step of this method.
b) Write a Matlab function that computes $u_1, \ldots, u_N$, by calling the function RK.

c) Use your implementation to approximate a solution of the initial value problem

\[
\begin{align*}
y'_1(t) &= y_2(t) \\
y'_2(t) &= -y_1(t) + y_2(t)(1 - y_1(t)^2).
\end{align*}
\]

Use the step size $\Delta t = 0.05$ over the time interval $[0, 100]$ and set, e.g., $y(0) = (0, 1)^T$.

d) Plot $y_2$ as a function of $y_1$. By looking at the plot, can you say something about the behavior of $y$ as time tends to infinity?