Fachbereich Mathematik & Informatik
Freie Universität Berlin
Prof. Dr. Ralf Kornhuber, Lasse Hinrichsen-Bischoff

Exercise 6 for the lecture

Numerical Mathematics II

WS 2021/2022

http://numerik.mi.fu-berlin.de/wiki/WS_2021/NumericsII.php

Due: 11:59pm on Monday, December 6, 2021

Problem 1

Consider the scalar differential equation

$$x' = \lambda(1 - x^2), \qquad \lambda > 0. \tag{1}$$

a) Show that $x_s^* = 1$ is an asymptotically stable and $x_u^* = -1$ an unstable fixed point of (1).

b) Compute the time step restriction for the explicit Euler method applied to the linearized problem

$$x' = -2\lambda(x - 1),$$

such that x_s^* is an asymptotically stable fixed point of the resulting discrete linear problem.

c) Is the time step restriction from b) also sufficient to guarantee that x_s^* is an asymptotically stable fixed point of the discrete nonlinear problem, resulting from applying the explicit Euler method to (1)?

Problem 2

We want to implement a general Runge–Kutta method for scalar ODEs of the form

$$x' = f(x), \qquad f : \mathbb{R} \to \mathbb{R}, \qquad f \in C^1(\mathbb{R}).$$

For implicit methods, the arising implicit equation should be solved by *Newton's* method.

a) Implement a method

```
runge_kutta(A, b, f, x, ff, tau)
```

which computes a single step of a Runge–Kutta method (starting at `x`, using step size `tau`) given by the Butcher scheme (`A`, `b`). Here, `f`$= f$ and `ff`$= f'$ (which you need for Newton's method).

b) Apply your method to the nonlinear equation from Problem 1 with $\lambda = 0.5$ and $\tau = 0.1$ in the interval $t \in [0, 10]$ using the following Butcher schemes:

   i) implicit Euler: $A = (1), b = (1)$,

   ii) implicit trapezoidal rule:

$$A = \begin{pmatrix} 0 & 0 \\ 1/2 & 1/2 \end{pmatrix}, \qquad b = \left(\frac{1}{2}, \frac{1}{2}\right).$$

   Try different starting values around x_s^* and x_u^* respectively. Can you confirm the stability results from Problem 1, a) numerically? Support your claims with corresponding plots!

Hint: For simplicity, you could apply a fixed number of Newton steps (e. g. 10 steps) instead of a more sophisticated stopping criterion.
Hint: For testing, you might want to test your code on a linear problem for which you know the solution. You could also compare against a direct implementation of the methods.