Heinrich Heine Universität Düsseldorf

Mathematisches Institut
Prof. Dr. Achim Schädle
Melinda Hagedorn

22.06.2026

Numerik gewöhnlicher Differentialgleichungen – 10. Übungsblatt

Aufgabe 33:

Bestimmen Sie die Stabilitätsbereiche

(a) der expliziten Mittelpunktsregel $y_{n+1} = y_{n-1} + 2hf_n$, sowie

(b) der Trapezregel $y_{n+1} = y_n + \frac{h}{2}(f_{n+1} + f_n)$

und skizzieren Sie beide in der komplexen Ebene.

Aufgabe 34:

Beweisen Sie Satz 3 aus (5.7):
Ist $\mathrm{ggT}(\rho, \sigma) = 1$ und ist $\sigma(\zeta) = 0$ für ein ζ mit $|\zeta| > 1$, so ist $\mathcal{S}$ beschränkt.

Aufgabe 35:

Automatisches Differenzieren ist eine Technik, um in einer Programmiersprache implementierte Funktionen nach den Eingabeparametern automatisch abzuleiten.

(a) Installieren Sie das Python Modul `jax`, siehe https://jax.readthedocs.io/en/latest/quickstart.html. Das geht am einfachsten mit `conda install jax -c conda-forge`.

(b) `Jax` erlaubt es, einfache Pythonfunktion automatisch abzuleiten. Um auch die meisten `numpy` Funktionen ableiten zu können, müssen Sie `numpy` durch `jax.numpy` ersetzen, wie zum Beispiel in

```
1 import jax.numpy as jnp
2 def pendel_jax(t, y):
3     erg = jnp.array([y[1], -jnp.sin(y[0])])
4     return erg
```

Erzeugen Sie mit Hilfe von `jacfwd` (oder `jacrev`) (`from jax import jacfwd, jacrev`) eine Pythonfunktion, die es erlaubt die Jacobimatrix bezüglich des Arguments `y` der Pythonfunktion `pendel` auszuwerten.

Allgemeine Informationen zum automatischen Differenzieren mit `jax`:
https://jax.readthedocs.io/en/latest/notebooks/autodiff_cookbook.html

Dokumentation zu `jacfwd`:
https://jax.readthedocs.io/en/latest/_autosummary/jax.jacfwd.html#jax.jacfwd

(c) Ersetzen Sie im impliziten Eulerverfahren aus Aufgabe 29, die Berechnung der Jacobimatrix durch eine Funktion, in der die Jacobimatrix durch automatisches Differenzieren berechnet wird.

(d) Wenden Sie nun die Verfahren BDF und Radau aus `scipy.integrate.solve_ivp` auf die Pendelgleichung

$$\ddot{\alpha}(t) = -\sin(\alpha(t)) \quad t \in [0, 3]$$

mit Startwerten $\alpha(0) = .1$ und $\dot{\alpha}(0) = 0$ an, wobei Sie die Jacobimatrix mit Hilfe von `jax` berechnen. Dazu können Sie den Befehl
`scipy.integrate.solve_ivp(pendel, t_span, y0, method='Radau', jac=jac_pendel)` bzw.
`scipy.integrate.solve_ivp(pendel, t_span, y0, method='BDF', jac=jac_pendel)`
verwenden, wenn Sie `jac_pendel` geeignet definieren.

Besprechung in der Übung am 29.06.2026.