Universität Konstanz
Fachbereich Mathematik und Statsitik
Johannes Schropp
Jan Rohleff

Sommersemester 2025

Universität Konstanz

Numerik gewöhnlicher Differentialgleichungen

https://www.math.uni-konstanz.de/~rohleff/numode.html

1. Übungsblatt

Ausgabe: 15.04.2025, Abgabe: 22.04.2025, bis 10.00 Uhr

Vorbemerkung zu den Programmieraufgaben

Die Programmieraufgaben können in Zweiergruppen bearbeitet werden. Dabei besteht die Option, die Aufgaben entweder in Matlab oder Python zu lösen. Die fertigen Programme sollten anschließend per E-Mail an jan.rohleff@uni-konstanz.de gesendet werden. Es ist wichtig, in den Programmen alle Schritte angemessen zu kommentieren, um die Nachvollziehbarkeit und Verständlichkeit zu gewährleisten.

Matlab

Für eine kurze Wiederholung zum Umgang mit Matlab kann unter folgendem Link das Skript *Einführung in Matlab* von E. Luik heruntergeladen werden:

https://www.math.uni-konstanz.de/~schropp/numsto/matlab.pdf

Python

Um schnelle numerische Berechnungen mit Python auszuführen, empfehlen wir die Open Source Erweiterung NumPy (Akronym für *Numerical Python*).

```
import numpy as np
```

Für eine kurze Wiederholung zum Umgang mit Python kann unter folgendem Link das Skript *Numerische Berechnungen mit Python* von S. Volkwein heruntergeladen werden:

https://www.mathematik.uni-konstanz.de/volkwein/python/python/

Aufgabe 1.1 (6 Punkte)

a) Zeigen Sie, dass durch $\Phi_t : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$, $t \in \mathbb{R}$

$$\Phi_t(x) = \left(e^t x_1, e^t x_2\right), \qquad \Phi_t(x) = \left(e^t x_1, e^{-t} x_2\right),$$
$$\Phi_t(x) = \left(x_1, t x_1 + x_2\right), \quad \Phi_t(x) = \left(e^{-t} x_1, (t x_1 + x_2) e^{-t}\right)$$

globale Flüsse auf $\mathbb{R}^2$ definiert werden und skizzieren Sie für verschiedene $x_0 \in \mathbb{R}^2$ die zugehörigen Phasenbilder.

b) Sei $\Phi_t : \mathbb{R}^N \to \mathbb{R}^N$ $(t \in \mathbb{R})$ ein globaler Fluss und $\gamma(x_0) = \{\Phi_t(x_0) : t \in \mathbb{R}\}$ der zu x_0 gehörige Orbit. Zeigen Sie, dass zwei Orbits $\gamma(x_0)$ und $\gamma(\tilde{x}_0)$ entweder identisch oder disjunkt sind.

Aufgabe 1.2 (6 Punkte)

Gegeben seien eine offene und konvexe Menge $\Omega \subset \mathbb{R}^N \times \mathbb{R}^N$ und die Differentialgleichung

$$\dot{x} = f(x), \quad f \in C^1(\Omega, \mathbb{R}^{2N}). \tag{1}$$

a) Das System (1) sei **hamiltonsch**, d. h. für $x = (q, p)$ habe (1) die Form

$$\dot{q} = \frac{\partial H}{\partial p}(q,p), \quad \dot{p} = -\frac{\partial H}{\partial q}(q,p) \tag{2}$$

mit einem $H \in C^2(\Omega, \mathbb{R})$. Der Lösungsfluss von (2) zum Anfangswert $(q_0, p_0) \in \Omega$ sei $\Phi_t(q_0, p_0)$.
Zeigen Sie: Für alle $t \in \mathbb{R}$, für die der Lösungsfluss Φ_t definiert ist, ist $\Phi_t : \Omega \to \mathbb{R}^{2N}$ eine symplektische Abbildung.
Definition: Eine differenzierbare Abbildung $g : \Omega \to \Omega$ heißt symplektisch, falls

$$Dg(q,p)^T J Dg(q,p) = J$$

für $(q, p) \in \Omega$ mit $J = \begin{pmatrix} 0 & I_N \\ -I_N & 0 \end{pmatrix} \in \mathbb{R}^{2N \times 2N}$ gilt.

b) Die Jacobi-Matrix $Df(x)$ sei symmetrisch für alle $x \in \Omega$. Zeigen Sie, dass es eine Funktion $H : \Omega \to \mathbb{R}$ gibt mit

$$f(x) = \nabla H(x), \quad x \in \Omega.$$

Hinweis: Definieren Sie $H(x) := \int_0^1 g(t,x)\,dt$ für eine geeignete Funktion g und zeigen Sie damit, dass für $k = 1, \ldots, 2N$ gilt: $\frac{\partial}{\partial x_k} g(t,x) = \frac{d}{dt}\left(t f_k(x_0 + t(x - x_0))\right)$, wobei $x_0 \in \Omega$ beliebig.

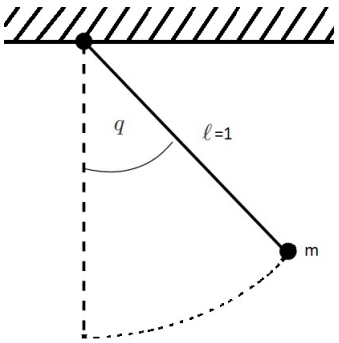
c) Zeigen Sie: Die Differentialgleichung (1) ist genau dann ein hamiltonsches System, wenn der zugehörige Lösungsfluss symplektisch ist.

Aufgabe 1.3 (9 Punkte)

Gegeben sei die Hamiltonfunktion des mathematischen Pendels der Länge 1

$$H(q,p) = \frac{1}{2m}p^2 - mg\cos(q),$$

wobei m der Massepunkt und g die Erdbeschleunigung darstellen.



a) Leiten Sie aus H die zugehörige Differentialgleichung (Hamiltonsystem) für q und p her. Dabei ist q der oben eingezeichnete Winkel und p die Winkelgeschwindigkeit.

b) Implementieren Sie in MATLAB oder Python sowohl das **explizite Euler-Cauchy-Verfahren** als auch das **implizite Euler-Cauchy-Verfahren** zur numerischen Lösung der Differentialgleichung:

$$\dot{x}(t) = f(t, x(t)), \quad x(0) = x_0,$$

mit der Schrittweite $h > 0$.

Die explizite und implizite Euler-Verfahren sind wie folgt definiert:

- **Explizites Euler-Cauchy-Verfahren:**

$$x_{k+1} = x_k + hf(t_k, x_k), \quad k = 0, 1, 2, \ldots$$

- **Implizites Euler-Cauchy-Verfahren:**

$$x_{k+1} = x_k + hf(t_{k+1}, x_{k+1}), \quad k = 0, 1, 2, \ldots$$

Wenden Sie diese Verfahren auf die Hamilton-Gleichungen aus Teil a) an. Die Anfangswerte sind:

$$q(0) = \frac{\pi}{4}, \quad p(0) = 0,$$

und das Zeitintervall ist $[0, 5]$ mit der Schrittweite $h = 0.01$. Wählen Sie für den Massepunkt $m = 1$ und die Erdbeschleunigung $g = 9.81$.

c) Zeichnen Sie die numerischen Lösungen sowohl in einem **Zeitdiagramm** (mit der Zeit auf der x-Achse und den Werten von q und p auf der y-Achse) als auch in einem **Phasenbild** (mit p auf der y-Achse und q auf der x-Achse).

Vergleichen Sie die numerischen Lösungen der beiden Verfahren (explizit und implizit) mit der Referenzlösung des MATLAB-Solvers `ode45` (oder der entsprechenden Lösung in Python, wobei wir hier die `scipy`-Bibliothek empfehlen).

Hinweise:

- Das beim impliziten Euler-Cauchy-Verfahren auftretende nichtlineare Gleichungssystem soll in jedem Zeitschritt mit dem Befehl `fsolve` (in MATLAB) oder einer numerischen Methode in Python gelöst werden. Verwenden Sie als Startwert die Lösung zum vorherigen Zeitpunkt.
- Für die Animation der Pendelbewegung ist der Befehl `drawnow` in MATLAB hilfreich. In Python kann dies durch `matplotlib.animation.FuncAnimation` umgesetzt werden.