Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Christian Engwer, MSc Marcel Koch

14.10.2019

Übung zur Vorlesung

Wissenschaftliches Rechnen

WS 2019/20 — Blatt 1

Abgabe: 21.10.2019, 10:00 Uhr, Briefkasten 112
Code zusätzlich per e-mail an `marcel.koch@uni-muenster.de`

Achtung: Achten Sie darauf, Ihre Programme ordentlich zu formatieren und gut zu kommentieren. Die Form wird mit in die Bewertung eingehen.

Aufgabe 1 (Energieerhaltung der Pendelgleichung) (4 Punkte)

Neben der Konsistenzordnung können auch Erhaltungseigenschaften von Verfahren für die Qualität einer numerischen Lösung wichtig sein. Wir betrachten die *allgemeine Pendelgleichung*

$$my''(t) + k(y(t)) = 0, \quad y(0) = y^0, \quad y'(0) = v^0 \tag{1}$$

mit nichtlinearer Rückstellkraft $k : \mathbb{R} \to \mathbb{R}$, Masse $m \in \mathbb{R}^+$, Anfangsauslenkung $y^0 \in \mathbb{R}$ und Anfangsgeschwindigkeit $v^0 \in \mathbb{R}$.

(a) Zeigen Sie, dass im linearen Fall $k(y) = \kappa y$, $\kappa \in \mathbb{R}^+$, für die Lösung der Pendelgleichung die Gesamtenergie

$$E(y, y') := \frac{1}{2} m (y')^2 + \frac{1}{2} \kappa y^2$$

in der Zeit konstant bleibt.

(b) Betrachten Sie das Leap-Frog-Verfahren zur Diskretisierung des autonomen AWPs (1) im linearen Fall. Zeigen Sie, dass die Gesamtenergie näherungsweise erhalten bleibt, d.h.

$$E\big[y(t^{n+1}), y'(t^{n+1})\big] - E\big[y(t^n), y'(t^n)\big] = \mathcal{O}(\Delta t^3)$$

(c) Im nichtlinearen Fall ist die Gesamtenergie gegeben durch

$$E(y, y') := \frac{1}{2} m (y')^2 + \int_0^y k(\eta)\, d\eta.$$

Zeigen Sie, dass auch im nichtlinearen Fall für die Lösung der Pendelgleichung die Gesamtenergie in der Zeit konstant bleibt.

Aufgabe 2 (Symplektizität) (2 Punkte)

Wir betrachten den linearen Fall der Pendelgleichung (1). Zeigen Sie, dass das Heun-Verfahren angewandt auf das zu dieser Gleichung äquivalente AWP 1. Ordnung nicht symplektisch ist.

Tipp: In der Vorlesung wurde gezeigt, dass das explizite Euler-Verfahren angewandt auf das dort betrachtete (entdimensionalisierte) mathematische Pendel nicht symplektisch ist. Sie können analog zur Vorlesung vorgehen. Überlegen Sie sich zunächst, wie der numerische Fluß $\Phi_{heun,\Delta t} : (y^k, v^k) \mapsto (y^{k+1}, v^{k+1})$ aussieht.

Aufgabe 3 (Explizites und Implizites Euler-Verfahren) (3 Punkte)

Implementieren Sie zwei Klassen `[Explicit|Implicit]EulerScheme` zur Durchführung beider Euler-Verfahrens für den allgemeinen Fall eines nicht-autonomen AWPs 1. Ordnung. Beide Klassen sollen dabei ein gemeinsames Interface erfüllen, das wie folgt definiert ist:

- Der Konstruktor beider Klassen erhält eine Funktion $f : I \times \mathbb{R}^n \to \mathbb{R}^n$ und im Falle des impliziten Euler-Verfahrens zusätzlich eine Funktion $J : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ zur Berechnung der Jacobi-Matrix von f.
- Mit der Methode

  ```
  def apply (t, dt, y_old)
  ```

  sollen die durch die Iterationsvorschrift festgelegten Schritte des Verfahrens ausgeführt werden können. Dabei bezeichnen `t` sowie `dt` den Zeitpunkt t_k sowie die Schrittweite Δt des aktuellen Zeitschritts und `y_old` entspricht der Approximationen y_k. Rückagbewert dieser Funktion ist die Approximation zum neuen Zeitpunkt y_{k+1}.

  Das für das implizite Euler-Verfahren benötigte Lösen eines Gleichungssystem kann von bereits existierenden python Paketen übernommen werden. Zum Beispiel eignet sich die Funktion *root*[1] des Pakets *scipy.optimization*[2] dafür.

Aufgabe 4 (Mathematisches Pendel, Computermodell) (5 Punkte)

$$\varphi''(t) + \frac{g}{l}\sin(\varphi(t)) = 0 \qquad \text{auf} \quad I := [0, T], \qquad T \in \mathbb{R}^+, \tag{2a}$$

$$\varphi(0) = \varphi_0, \qquad \varphi_0 \in \mathbb{R}, \tag{2b}$$

$$\varphi'(0) = v_0, \qquad v_0 \in \mathbb{R}. \tag{2c}$$

Um das mathematische Pendel (2) zu simulieren, wollen wir für dieses ein Computermodell herleiten. Mit dem Ansatz $\varphi'(t) = v(t)$ lässt sich (2) in ein System von zwei gewöhnlichen Differentialgleichungen 1. Ordnung mit zugehörigen Anfangswerten transformieren:

$$\varphi'(t) = v(t), \qquad v'(t) = a \qquad \text{auf} \quad I, \tag{3a}$$

$$\varphi(0) = \varphi_0, \qquad v(0) = b. \tag{3b}$$

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html#scipy.optimize.root`

[2] `https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html`

(a) Bestimmen Sie die fehlenden Terme a und b. Bestimmen Sie das resultierende AWP (3) für das explizite und implizite Euler-Verfahren. Schreiben Sie zusätzlich das Gleichungssystem auf, das beim impliziten Eulerverfahren gelöst werden muss.

(b) Implementieren Sie beide resultierende Computermodell in python. Benutzen Sie dazu Ihre Implementierungen aus Aufgabe 3. Speichern Sie für jeden Zeitschritt t_k den dazu gehörigen Auslenkungswinkel und Geschwindigkeit für eine spätere Visualisierung.

(c) Führen Sie mit Ihrem Programm verschiedene Simulationen durch und visualisieren Sie die Ergebnisse. Für die Darstellung eignet sich das python Paket *matplotlib*[3]. Benutzen Sie dabei den Anfangswert $v_0 = 0$ und die maximale Zeit $T = 12.0$.

   (i) Wählen Sie $\varphi_0 = 0.5$ als festen Anfangswert.
   Wählen Sie verschiedene Schrittweiten $\Delta t = 2^{-i},\ \ i = 4, 5, ..., 10$.
   Plotten Sie den Auslenkungswinkel, die Geschwindigkeit und die gesamte Energie Ihrer Simulation. Die Daten der verschiedenen Schrittweiten sollen in einer gemeinsamen Grafik dargestellt werden. Erörtern Sie Ihre Ergebnisse.

   (ii) Wählen Sie die beiden verschiedenen Anfangswerte $\varphi_0 = 0.5$ und $\varphi_0 = 1.0$.
   Wählen Sie $\Delta t = 2^{-15}$ als feste Schrittweite.
   Plotten Sie den Auslenkungswinkel, die Geschwindigkeit und die gesamte Energie Ihrer Simulation. Die Daten der verschiedenen Schrittweiten sollen in einer gemeinsamen Grafik dargestellt werden. Erörtern Sie Ihre Ergebnisse.

[3] `https://matplotlib.org/`