ETHZ, D-MATH

Prüfung
Numerische Methoden D-PHYS, FS 2018

Dr. V. Gradinaru

15.08.2018

1. *(8 Punkte) Nichtlineares System*

Das nichtlineare Gleichunssystem

$$e^{xy} + x^2 + 2y - \frac{8}{14} = 0 \tag{1}$$

$$x^2 + y^2 + 3x - \frac{14}{8} = 0 \tag{2}$$

ist zu lösen.

a) Implementieren Sie hierzu ein Newton-Verfahren in `newton(x0, F, DF, rtol, maxiter)` in `non_linear_system.py`, wobei

- `x0` die Initialwert,
- `F` die Funktion F
- `DF` ist DF
- `rtol` die relative Toleranz
- `maxiter` maximal Anzahl Iterationen

Die Funktion `newton` soll ein Tupel `x,i, x_sequence` zurückgeben, wobei

- `x` die approximierte Lösung
- `i` Anzahl Iterationen
- `x_sequence` die Folge $\{x_i\}$ von dem Newton-Verfahren

b) Lösen Sie mit der Funktion `newton` das obige System mit den Startwerten $x = \frac{3}{5}$ und $y = \frac{1}{2}$ bis auf eine Toleranz von 10^{-14}.

Benutzen Sie das Template `non_linear_system.py`.

c) Untersuchen Sie die beobachteten Konvergenzordnungen für folgende Startwerte:

(i) $x = \frac{3}{5}$ und $y = \frac{1}{2}$
(ii) $x = \frac{2}{5}$ und $y = \frac{1}{4}$
(iii) $x = -\frac{23}{5}$ und $y = \frac{41}{5}$.

Vergleichen Sie die beobachtete Konvergenzordnung mit der theoretischen.

Bitte wenden!

2. *(8 Punkte) Lobatto-Quadratur*

Lobatto-Quadraturformeln besitzen die maximale Ordnung unter jenen Quadraturformeln, die die Intervallendpunkte als Knoten haben:

$(b_i, c_i)_{i=1}^s$ mit $c_1 = 0$, $c_s = 1$ der Ordnung $2s - 2$.

a) Geben Sie die Lobatto-Quadraturformeln der Ordnungen 2 und 4 an ($s = 2$ und $s = 3$).

b) Berechnen Sie die Knoten und Gewichte der Lobatto-Quadraturformel der Ordnung 6 ($s = 4$).

Hint: : Die Berechnung vereinfacht sich, wenn Sie verwenden, dass die gesuchte Quadraturformel symmetrisch ist.

Siehe nächstes Blatt!

3. *(8 Punkte) QR-Zerlegung*

Sei die Matrix A mit den Spaltenvektoren $[1, 1, 1, 1]^T$ und $[-2, 0, 1, 3]^T$.

a) Berechnen Sie auf Papier die zwei Spiegelungsmatrizen, die die QR-Zerlegungvon A realisieren und geben Sie die Faktoren Q und R an.

b) Berechnen Sie auf Papier die QR-Zerlegung von A mittels einer Spiegelung und zwei Drehungen.

Bitte wenden!

4. *(8 Punkte) Methode der kleinsten Quadrate*

Seien $h_i \in \mathbb{R}$ die Messungen des Herzgewichtes einer Katze mit Körpergewicht k_i (für $i = 1, \ldots, m$). Das mathematische Model ist

$$h = ak + c, \qquad \text{hier: } h \text{ die Herzgrösse, und } k \text{ die Körpergösse,} \tag{3}$$

a) Schreiben Sie die Methode der kleinsten Quadrate zu diesem Model mit Matrix **A** und rechte Seite **b** und mit der Hilfe von h_i und k_i.

b) Wenn $\mathbf{x} = \begin{pmatrix} x_1, \cdots, x_n \end{pmatrix}$ die Lösung im Sinne der kleinsten Quadrate von **a)** ist, was sind a und c in (3)?

c) Implementieren Sie die Funktion `A_berechnen(k)` in `katze.py`, wobei k ein `numpy.array` mit den gemessenen Körpergewichten ist.

Die Funktion sollte die Matrix **A** zurückgeben.

d) Implementieren Sie die Funktion `b_berechnen(h)` in `katze.py`, wobei: h ein `numpy.array` mit den gemessenen Herzgewichten ist.

Die Funktion sollte den Vektor **b** zurückgeben.

e) Implementieren Sie die Funktion `kleinsten_quadrate(k, h)` in `katze.py`, wobei:

- k ein `numpy.array` mit den gemessenen Körpergewichte
- h ein `numpy.array` mit den gemessenen Herzgewichte
- `k.shape == h.shape`

Die Funktion `kleinsten_quadrate(k, h)` sollte a und c zurückgeben.

Siehe nächstes Blatt!

5. *(8 Punkte) Splitting*

Sei $\epsilon = 0.001$, $\omega = 0.01$ und die Differentialgleichung

$$\begin{cases} \ddot{u} = -u + \epsilon \cos(\omega t) \\ u(0) = 1, \dot{u}(0) = 0 \end{cases} \tag{4}$$

a) **Schreiben** Sie die entsprechende **Hamilton-Funktion**

b) **Implementieren** Sie die Funktion `strang_splitting` in `strang_splitting.py`. Die Funktion sollte (4) mit einem **Strang Splitting Verfahren** lösen.

Die Argumente sind

- `u0` der Initialwert $u(0)$
- `du0dt` der Initialwert $\dot{u}(0)$
- `t_end` die Finale Integrationszeit T
- `n_steps` Anzahl Zeitschritte

Die Funktion sollte ein Tupel `t, u` zurückgeben, in dem `t` die Zeitschritte sind, und `u` die Approximierte Lösung ist.

Hint: Zur Verfügung steht die Funktion `splitting_method`.

c) Plotten Sie die Lösung bis $T = 10$. Finden Sie durch ausprobieren einen T bei dem die Methode mit `n_steps=4000` offensichtlich eine falsche Lösung produziert und erklären Sie (auf Papier) warum das passiert.

Hint: Zur Verfügung steht die Funktion `run_and_plot`.

Bitte wenden!

6. *(8 Punkte)Exponentielles Euler-Verfahren*

Betrachten Sie das *exponentielle Euler-Verfahren* mit konstanter Schrittweite:

$$\vec{y}_{k+1} = \vec{y}_k + h\varphi(h\vec{J}_f)f(\vec{y}_k), \quad k = 0, \ldots, N \tag{5}$$

wobei:

$$\vec{J}_f := \mathrm{D}\, f(\vec{y}_k), \quad \varphi(z) = \frac{e^z - 1}{z}$$

a) Leiten Sie die Stabilitätsfunktion $S(z)$ aus (5) her.

b) Schreiben Sie eine `Python`-Funktion `expEV`, die das nichtlineare Anfangswertproblem (4) von Aufgabe 4 mit dem exponentiellen Eulerverfahren (5) mit konstanter Schrittweite löst.

Die Argumente sind:

- `u0` der Initialwert $u(0)$
- `du0dt` der Initialwert $\dot{u}(0)$
- `t_end` die Finale Integrationszeit T
- `n_steps` Anzahl Zeitschritte

Hinweis: Verwenden Sie das Template `exp_euler.py`.

Hinweis: $Df \in \mathbb{R}^{2\times 2}$ ist klein . Sie können also hier einfach `expm` verwenden.

c) Plotten Sie die Lösung bis $T = 10$ und $T = 9000$, wobei `n_steps=4000` sein soll. Erklären Sie kurz was passiert.