ETHZ, D-MATH

# Probeprüfung

# Numerische Methoden D-PHYS, FS 2019

Dr. V. Gradinaru

29.03.2019

Auf der Vorlesungshompage stehen Tamplates für die Probeprüfung bereit.
Prüfungsdauer : 90 Minuten.

**1.** *(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.

**c)** Sei die symmetrische Quadraturformel auf $[-1, 1]$ gegeben durch die Knoten $0, \pm\sqrt{\frac{5}{11} - \frac{2}{11}\sqrt{5/3}}, \pm\sqrt{\frac{5}{11} + \frac{2}{11}\sqrt{5/3}}, \pm 1$ und entsprechenden Gewichte $\frac{256}{525}, \frac{124+7\sqrt{15}}{350}, \frac{124-7\sqrt{15}}{350}, \frac{1}{21}$.

Finden Sie die Ordnung dieser Quadraturformel. Ist das auch eine Lobatto-Quadraturformel?

**Bitte wenden!**

**2.** *(8 Punkte) Splitting*

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

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

**a)** **Schreiben** Sie die Differentialgleichung (1) in einem System Differentialgleichungen erster Ordnung um:

$$\begin{cases} \dot{y}_1 = f_1(t, y_0, y_1) \\ \dot{y}_2 = f_2(t, y_0, y_1) \\ y_1(0) = 1 \\ y_2(0) = 0 \end{cases} \tag{2}$$

**b)** Implementieren Sie folgende Verfahren:

- Explizites Eulerverfahren
- Implizites Eulerverfahren
- Implizite Mittelpunktsregel

Alle Funktionen sollen die rechte Seite der ODE `rhs`, den Startwert `y0`, die Endzeit `T` und die Anzahl Schritte $N$ als Parameter akzeptieren, z.B:

```
1 def explict_euler(rhs, y0, T, N):
2     # Implementieren Sie hier das explizite Eulerverfahren.
```

*Hinweis 1.* Lösen Sie die auftretenden nicht-linearen Gleichungssysteme $F(y) = 0$ mit `scipy.optimize.fsolve`.

*Hinweis 2.* Für diese Aufgabe können Sie `ode_solvers.py` verwenden.

**c)** Testen Sie Ihren Code mit Endzeit $T = 10$ und $N = 100$ und ploten Sie die entsprechenden Lösungen $u$ in einem Bild.

*Hinweis 2.* Für diese Aufgabe können Sie `Differential.py` verwenden.

**d)** Bestimmetn Sie die Konvergenzordnung von

$$|y_N - y(T)| \tag{3}$$

mit $T = 1$. Betrachten Sie dazu die Folge der Schrittenanzahl $N_k = 2^k$, $k \in \{4, \ldots, 11\}$ und für jede $k$, berechnen Sie $|u_{N_k} - y(T)|$. Erstellen Sie ein doppelt logarithmisches Plot des Fehlers. Vergleichen Sie diese mit der theoretischen Konvergenzordnung.

*Hinweis.* Benutzen Sie `ode45.py` um eine hervorragende Approximation von $y(T)$ zu berechnen. Sie finden `ode45` im Template `ode45.py`.