D-MATH
Dr. Vasile Gradinaru
Marco Petrella

Serie 3

Abgabedatum: Di. 12.3 / Mi. 13.3 oder früher
Koordinatoren: Marco Petrella, HG J 45, `marco.petrella@sam.math.ethz.ch`
Webpage: `http://metaphor.ethz.ch/x/2019/fs/401-1662-10L`

1. Kernaufgabe: Gauss-Legendre Quadratur und Golub-Welsch Algorithmus

Aufgabenstellung

a) Leiten Sie die Drei-Term-Rekursion her, welche die Legendre Polynome $P_n(x)$ beschreibt. Das Skalarprodukt ist in diesem Fall gegeben als:

$$\langle p, q \rangle := \int_{-1}^{1} p(x)q(x)\mathrm{d}x$$

und es gilt $P_0(x) = 1$ und $P_1(x) = x$.

b) Begründen Sie die Wahl der Einträge in der Jacobi-Matrix **J** im Algorithmus von Golub-Welsch. (Siehe Code 1.3.3 im Skript.)

c) Verwenden Sie folgende Quadraturregeln

- zusammengesetzte Trapezregel
- zusammengesetzte Simpsonregel
- Gauss-Legendre Quadratur
- zusammengesetzte Gauss-Legendre Quadratur

um das Integral

$$I = \int_{0}^{1} f_i(x)\mathrm{d}x$$

von $f_i(x)$ auf N Teilintervallen oder mit n Funktionsauswertungen zu berechnen. (Die genauen Werte von N und n stehen im Template.) Die beiden Funktionen sind gebeben durch

$$f_1(x) := \frac{1}{1+5x^2} \qquad f_2(x) := \sqrt{x}\,.$$

Berechnen Sie den Fehler und plotten Sie die Konvergenzraten. Welche Methode verwendet man sinnvollerweise?

Hinweis: Verwenden Sie das Template `3_gw_Template.py`

d) Wie viele Funktionsauswertungen braucht die h-adaptive Strategie von Code 1.4.2 für die beiden Funktionen $f_i(x)$? Plotten Sie die erzeugten Gitter.

Hinweis: Modifizieren Sie den Code so, dass er alle erzeugten Gitter zurück gibt.

Bitte wenden!

2. *Pendelgleichung*

Wir betrachten die Gleichung für den Auslenkungswinkel $\alpha(t)$ im mathematischen Pendel der Länge l im Erdgraviationsfeld (wir notieren $\omega^2 = \frac{g}{l}$):

$$\begin{aligned} \ddot{\alpha}(t) &= -\omega^2 \sin \alpha(t) \\ \alpha(0) &= \alpha_0 \\ \dot{\alpha}(0) &= \dot{\alpha}_0 \end{aligned} \tag{1}$$

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

$$\dot{y}_0 = f_0(y_0, y_1) \tag{2}$$
$$\dot{y}_1 = f_1(y_0, y_1) \tag{3}$$
$$\begin{aligned} y_0(0) &= \alpha_0 \\ y_1(0) &= \dot{\alpha}_0 \,. \end{aligned}$$

Für welche physikalische Grösse steht hier y_1?

b) Skizzieren Sie Ihren Code auf Papier. Benutzten Sie diese Skizze um den Code zu planen. Sie sollten sich überlegen welche Bausteine in den ODE-Lösern verwendet werden. Wie unterscheidet sich das explizite Eulerverfahren vom impliziten Verfahren. Was haben die beiden gemeinsam? Wie passt das velocity-Verlet Verfahren zu Ihren Bausteinen? Vergleichen Sie Ihre Struktur mit der Stuktur der Templates, besonders `ode_solvers.py`.

c) Implementieren Sie folgende Verfahren:

- Explizites Eulerverfahren
- Implizites Eulerverfahren
- Implizite Mittelpunktsregel
- Velocity-Verlet Verfahren

Alle Funktionen sollen die rechte Seite der ODE, den Startwert, die Endzeit und die Anzahl Schritte als Parameter akzeptieren, z.B:

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

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

Hinweis 2. Auf der Vorlesungshomepage stehen Templates für die Serie bereit. Für diese Aufgabe können Sie `ode_solvers.py` verwenden.

d) Testen Sie Ihren Code für $g = 9.81$, $l = 0.6$, $T = 0.2$.. Bestimmen Sie die Konvergenzordnung von

$$|y_N - y(T)| \tag{4}$$

und vergleichen Sie diese mit der theoretischen Konvergenzordnung.

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

Siehe nächstes Blatt!

e) Implementieren Sie den konkreten Fall $g = 9.81$, $l = 0.6$, $T = 4.0$, $N = 500$, $\alpha_0 = 1.4855$, $\dot{\alpha}_0 = 0$. Ploten Sie alle Trajektorien in einem Bild mit den Koordinaten $\alpha, \dot{\alpha}$. Ploten Sie ein Bild der Evolution der (Approximationen der) kinetischen ($\frac{1}{2}\dot{\alpha}(t)^2$), potentiellen ($\omega^2(1 - \cos(\alpha(t)))$) und totalen Energie im System (Energie gegen Zeit) für jede Methode. Bleibt die totale Energie immer erhalten? Stimmen die vorausgesagten Perioden überein? Welche liegt der Realität am nächsten?

Hinweis. Das Template für diese Aufgabe ist `pendulum.py`

f) Messen Sie die Laufzeit der vier Methoden. Welche Methode ist am effizientesten?