Dr. Vasile Gradinaru
Luc Grosheintz

Serie 2

Abgabedatum: Di. 14.3 / Mi. 15.3 oder früher, in den Übungsgruppen
Koordinatoren: Luc Grosheintz, HG J 46, luc.grosheintz@sam.math.ethz.ch
Webpage: http://metaphor.ethz.ch/x/2017/fs/401-1662-10L

1. *Neues in* Python

In Python können Sie Funktionen als Funktionsargumente übergeben.

```
import numpy as np

def apply(f, x):
    return f(x)

def square(x):
    return x*x

x = np.random.random((3, 4))

apply(print, x)
print(apply(np.sin, x))
print(apply(lambda x: np.sum(x, axis=-1), x))
print(apply(square, x))
```

Einen Graph erstellen Sie wie folgt

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(1e-5, 1.2, 10000)
fx = np.sin(2.0*np.pi/x)
plt.plot(x, fx, label="sin")
plt.legend()
plt.xlabel("x-Label")
plt.ylabel("y-Label")

plt.savefig("sin.png")
plt.savefig("sin.eps")
plt.show()
```

Bitte wenden!

2. *Pendelgleichung*

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

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

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

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

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 Bausteine? Vergleichen Sie Ihre Struktur mit der Stuktur der Templates, besonder `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

$$|y_N - y(T)| \qquad (4)$$

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 ($-\omega^2 \cos(\alpha(t))$) 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?

3. Kernaufgabe: Das N-Körper Problem

Modellierung der Physik

Wir betrachten N Körper im dreidimensionalen Raum. Ihre Dynamik unterliegt einzig der Gravitation, beschrieben durch das Newtonsche Gesetz. Jeder Körper K_i hat eine Position $\underline{q_i} \in \mathbb{R}^3$ und einen Impuls $\underline{p_i} \in \mathbb{R}^3$ sowie eine Masse m_i. Zur einfachen Behandlung der N Körper fassen wir deren Positionen $\underline{q_i}$ und Impulse $\underline{p_i}$ in Vektoren zusammen:

$$\underline{q} := [\underline{q_1}| \dots |\underline{q_N}]^{\mathrm{T}} \in \mathbb{R}^{3N}$$
$$\underline{p} := [\underline{p_1}| \dots |\underline{p_N}]^{\mathrm{T}} \in \mathbb{R}^{3N} .$$

Die Hamilton-Funktion $\mathcal{H}(\underline{q}, \underline{p})$ lautet dann:

$$\mathcal{H}\left(\underline{q}, \underline{p}\right) = \frac{1}{2} \sum_{i=1}^{N} \frac{1}{m_i} \underline{p_i}^{\mathrm{T}} \underline{p_i} - G \sum_{1 \leq i < j \leq N} \frac{m_i m_j}{\|\underline{q_j} - \underline{q_i}\|}$$

Aus dieser Funktion erhält man die Bewegungsgleichungen durch partielles ableiten:

$$\frac{\partial \underline{q}}{\partial t} =: \underline{\dot{q}} = \frac{\partial \mathcal{H}}{\partial \underline{p}}$$
$$\frac{\partial \underline{p}}{\partial t} =: \underline{\dot{p}} = -\frac{\partial \mathcal{H}}{\partial \underline{q}} .$$

Wir finden hier also:

$$\underline{\dot{q}_k} = \frac{\partial \mathcal{H}}{\partial \underline{p_k}} = \frac{\partial}{\partial \underline{p_k}} \frac{1}{2} \sum_{i=1}^{N} \frac{1}{m_i} \underline{p_i}^{\mathrm{T}} \underline{p_i} = \frac{1}{m_k} \underline{p_k}$$

und:

$$\underline{\dot{p}_k} = -\frac{\partial \mathcal{H}}{\partial \underline{q_k}} = \frac{\partial}{\partial \underline{q_k}} G \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{m_i m_j}{\|\underline{q_j} - \underline{q_i}\|} = G \sum_{\substack{i=1 \\ i \neq k}}^{N} \frac{m_i m_k}{\|\underline{q_i} - \underline{q_k}\|^3} \left(\underline{q_i} - \underline{q_k}\right) .$$

Schliesslich fasst man noch die Positionen $\underline{q}$ und Impulse $\underline{p}$ in einen einzigen grossen Vektor $\underline{y} := [\underline{q}|\underline{p}]^{\mathrm{T}}$ zusammen.

Mathematisches Modell

Aus obiger Herleitung bekommt man ein System von gekoppelten Differentialgleichungen erster Ordnung:

$$\underline{\dot{y}}(t) = \begin{bmatrix} \underline{\dot{q}}(t) \\ \underline{\dot{p}}(t) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial \underline{p}} \\ -\frac{\partial \mathcal{H}}{\partial \underline{q}} \end{bmatrix} = f\left(t, \underline{y}\right)$$

mit $f : \mathbb{R}^+ \times \mathbb{R}^{6N} \to \mathbb{R}^{6N}$ welches wir nun mit den Polygonzugverfahren aus der Vorlesung lösen wollen.

Siehe nächstes Blatt!

Aufgabenstellung

a) Implementieren Sie mithilfe obiger Formeln die Funktion $f(\underline{y})$ korrekt für N Körper.

Hinweis: Benutzen Sie das `Python` Template `nbody.py`

b) Implementieren Sie eine mit dem velocity-Verlet Verfahren kompatible Version der rechten Seite. Achten Sie dabei auf die Verwendung korrekter Input-Werte.

c) Testen Sie die Implementation am Zweikörperproblem mit folgenden Anfangswerten. Massen: $m_1 = 500$, $m_2 = 1$, Positionen: $\underline{q_1} = \underline{0}$, $\underline{q_2} = (2, 0, 0)$, Impulse: $\underline{p_1} = \underline{0}$, $\underline{p_2} = (0, \sqrt{\frac{Gm_1}{2}}, 0)$ und $G = 1$. Die Endzeit sei $T = 3$ und es sollen 5000 Zeitschritte gemacht werden. Plotten Sie die Bahnen der Körper in der x-y Ebene. Vergleichen Sie die benötigten Rechenzeiten der verschiedenen Methoden.

d) Als nächstes wollen wir ein Dreikörperproblem betrachten. Im Allgemeinen ist das Dreikörperproblem nicht analytisch lösbar. Untersuchen Sie numerisch die Dynamik mit den gegebenen Anfangswerten. Massen: $m_1 = m_2 = m_3 = 1$. Positionen und Impulse:

$$\begin{aligned} \underline{q_1} &= (0.97000436, -0.24308753, 0) & \underline{p_1} &= (0.46620368, 0.43236573, 0) \\ \underline{q_2} &= (-0.97000436, 0.24308753, 0) & \underline{p_2} &= (0.46620368, 0.43236573, 0) \\ \underline{q_3} &= (0, 0, 0) & \underline{p_3} &= (-0.93240737, -0.86473146, 0)\,. \end{aligned}$$

Auch hier nehmen wir $G = 1$. Die Endzeit sei $T = 2$ und es sollen 1000 Zeitschritte mit dem velocity-Verlet impliziten Mittelpunktsregel gemacht werden. Plotten Sie die Bahnen der drei Körper in der x-y Ebene.[a]

e) Abschliessend wollen wir die Bahnen der Planeten[b] des äusseren Sonnensystems untersuchen. Dazu integrieren wir ihre Bewegungsgleichungen ausgehend von konkreten Anfangswerten. Diese Werte sind im `Python` Template zur Aufgabe notiert und alle Einheiten sind im astronomischen Einheitensystem (Längen in A.U., Massen relativ zur Sonne und Zeit in Tagen). Hier nehmen wir $G = 2.95912208286 \cdot 10^{-4}$. Die Endzeit sei $T = 20000$ und es sollen mit jeder Methode 2000 Zeitschritte gemacht werden. Plotten Sie die Planetenbahnen in der x-y Ebene. (Die Berechnung mit den impliziten Methoden dauert eine Weile.)

[a] Dieses sehr spezielle Bewegungsmuster ist als *figure-eight pattern* bekannt und detailliert beschrieben im Paper *A remarkable periodic solution of the three-body problem in the case of equal masses* von Alain Chenciner und Richard Montgomery, zu finden unter `http://arxiv.org/abs/math/0011268`

[b] Pluto ist offiziell zwar kein Planet mehr, wir wollen ihn hier dennoch berücksichtigen.