Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Dr. Stephan Rave

29. März 2017

Übung zum Kompaktkurs

Einführung in die Programmierung zur Numerik mit Python

Wintersemester 2016/17 — Hausaufgabe

Abgabe Ihres Programms als Mailanhang mit vollständigen Namen an stephan.rave@wwu.de bis spätestens Mittwoch, 5. April 2017, 17:00 Uhr

Aufgabe 1 (Mehrkörperproblem)

Wir betrachten das Mehrkörperproblem für N Körper. Die Körper sind durch eine Punktmasse in $\mathbb{R}^3$ eine Position und eine Geschwindigkeit definiert. Wir nehmen an, die Bewegung der Körper gehorchen den Differentialgleichungen

$$\dot{x}_i = v_i, \qquad i = 1, .., N, \tag{1a}$$

$$\dot{v}_i = G \sum_{i \neq j} \frac{m_j}{\|x_i - x_j\|_2^3}(x_i - x_j) \qquad i = 1, .., N. \tag{1b}$$

Diese sollen mit dem symplektischen und dem expliziten Euler-Verfahren angenähert werden. Hierbei wird aus einer Näherung (x_i^n, v_i^n) zum Zeitpunkt $t = \Delta t \cdot n$ eine neue Näherung (x_i^n, v_i^n) zum Zeitpunkt $t = \Delta t \cdot (n + 1)$ wie folgt bestimmt.

- Verfahrensvorschrift beim expliziten Euler-Verfahren:

$$x_i^{n+1} = x_i^n + \Delta t v_i^n, \qquad i = 1, .., N \tag{2a}$$

$$v_i^{n+1} = v_i^n - \Delta t G \sum_{i \neq j} \frac{m_j}{\|x_i^n - x_j^n\|_2^3}(x_i^n - x_j^n) \qquad i = 1, .., N \tag{2b}$$

- Verfahrensvorschrift beim symplektisches Euler-Verfahren:

$$x_i^{n+1} = x_i^n + \Delta t v_i^n, \qquad i = 1, .., N \tag{3a}$$

$$v_i^{n+1} = v_i^n - \Delta t G \sum_{i \neq j} \frac{m_j}{\|x_i^{n+1} - x_j^{n+1}\|_2^3}(x_i^{n+1} - x_j^{n+1}) \qquad i = 1, .., N \tag{3b}$$

Die Gravitationskonstante ist hierbei als $G = 2.95912208286 \times 10^{-4}$ und die Anfangswerte und Massen der Körper sind wie folgt gewählt:

- Beispiel 1 (Zwei Körper): Wählen Sie $\Delta t \approx 10^{-3}$ und den Endzeitpunkt $T \approx 10^5$.

Körper (Name)	Masse	Anfangsposition x_i^0	Anfangsgeschwindigkeit v_i^0
Tom	$m_1 = 200000$	(0,1,0)	(-1,0,1.5)
Jerry	$m_1 = 100000$	(0,-1,0)	(1,0,-1)

- Beispiel 2 (äußeres Sonnensystem): Wählen Sie $\Delta t \approx 10^1$ und den Endzeitpunkt $T \approx 2$.

Planet	Masse	Anfangsposition x_i^0	Anfangsgeschwindigkeit v_i^0
Jupiter	$m_1 = 0.00095478610403$	-3.5023653 -3.8169847 -1.5507963	0.00565429 -0.00412490 -0.00190589
Saturn	$m_2 = 0.000285583733151$	9.0755314 -3.0458353 -1.6483708	0.00168318 0.00483525 0.00192462
Uranus	$m_3 = 0.0000437273164546$	8.3101420 -16.2901086 -7.2521278	0.00354178 0.00137102 0.00055029
Neptun	$m_4 = 0.0000051779138449$	11.4707666 -25.7294829 -10.8169456	0.00288930 0.00114527 0.00039677
Pluto	$m_5 = 1.0/(1.3 \times 10^8)$	-15.5387357 -25.2225594 -3.1902382	0.00276725 -0.00170702 -0.00136504
Sonne	$m_6 = 1.00000597682$	(0,0,0)	(0,0,0)

Schreiben Sie ein `python` Programm namens *multibody.py* in dem Sie die beiden Beispiele von Mehrkörperproblemen mit explizitem und symplektischen Euler-Verfahren simulieren. Lassen Sie in diesem Programm die Laufbahnen der Körper plotten. Notieren Sie zusätzlichen Ihre Beobachtungen bzgl. der Unterschiede zwischen dem explizitem und symplektischen Euler-Verfahren. Betrachten Sie dafür unterschiedliche Zeitschritte.
Als Vorlage dient die Datei *multibody.py*, die Sie auf der Homepage zur Veranstaltung finden. Sie dürfen sich an dieser orientieren, müssen dies aber nicht tun. In der Vorlage finden Sie auch alle Konstanten und Anfangswerte aller Körper. In Abbildung 1 sehen Sie ein Beispielergebnis.

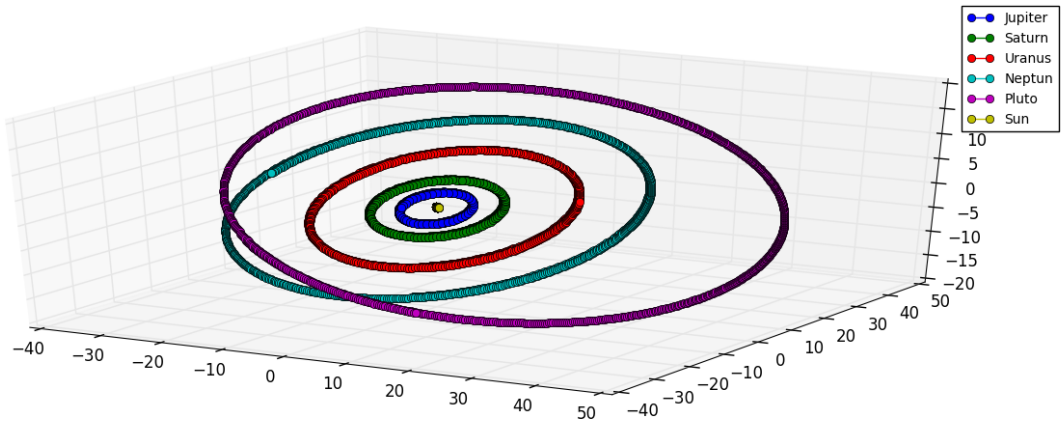

Abbildung 1: Beispielergebnis für das äußere Sonnensystem

Aufgabe 2 (Gauß-Algorithmus)

Sei $A \in \mathbb{R}^{n \times n}$ eine reelle Matrix und $b \in \mathbb{R}^n$ ein Vektor. Gesucht ist der Lösungsvektor $x \in \mathbb{R}^n$, für den gilt:

$$Ax = b$$

Aus der linearen Algebra kennen Sie den Gauß-Algorithmus mit Spalten-Pivot-Suche als ein Lösungsverfahren für ein solches lineares Gleichungssystem. Beim Pivoting wird im k-ten Schritt das betragsmäßig größte Element in der k-ten Spalte unterhalb der Diagonalen gesucht und die zugehörige Zeile anschließend mit der k-ten Zeile vertauscht.
Schreiben Sie ein `python` Programm namens $aufgabe1.py$ zur Realisierung des Gauß-Algorithmus. Verwenden Sie dabei `numpy.array` als grundlegende Datenstruktur für Vektoren als auch Matrizen. Gehen Sie in folgenden Schritten vor:

(a) Implementieren Sie eine Funktion, die das Pivoting durchführt, d.h. zu gegebenem k das Pivot-Element (und dessen Position) bestimmt.

(b) Schreiben Sie eine Funktion, die in einer gegebenen Matrix zwei Zeilen vertauscht.

(c) Implementieren Sie, aufbauend auf diesen zwei Funktionen, den Gauß-Algorithmus als Funktion, die zu gegebenem A, b das Gleichungssystem löst. Überprüfen Sie insbesondere, ob A und b vom richtigen Datentyp sind und ihre Größen richtig gewählt sind.

(d) Im Falle einer nicht invertierbaren Matrix soll Ihr Programm einen entsprechenden `ValueError` erzeugen. Überlegen Sie sich hierzu, woran man im Algorithmus diesen Fall erkennt.

Testen Sie das Verfahren an folgendem Gleichungssystem:

$$\begin{pmatrix} 2 & -1 & 5 & 7 \\ -1 & 1 & 1 & 4 \\ 1 & 2 & -3 & -2 \\ 3 & -1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 43 \\ 20 \\ -12 \\ 4 \end{pmatrix}$$