Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Christian Engwer, MSc Marcel Koch

21.10.2019

Übung zur Vorlesung

Wissenschaftliches Rechnen

WS 2019/20 — Blatt 1

Abgabe: 28.10.2019, 10:00 Uhr, Briefkasten 112
Code zusätzlich per e-mail an `marcel.koch@uni-muenster.de`

Achtung: Achten Sie darauf, Ihre Programme ordentlich zu formatieren und gut zu kommentieren. Die Form wird mit in die Bewertung eingehen.

Definition 1 (Fließkommazahlen)

$\mathbb{F}(\beta, r, s)$ sei die aus der Vorlesung bekannte allgemeine Darstellung von Fließkommazahlen mit Basis β, Anzahl r der Mantissenstellen und Anzahl s der Exponentenstellen.

Aufgabe 1 (Eigenschaften Fließkommazahlen) (3 Punkte)

Seien $X_-, X_+ \in \mathbb{R}$ die kleinste, bzw. größte darstellbaren Zahlen der $\mathbb{F}(\beta, r, s)$ und rd die in der Vorlesung eingeführte Rundungsfunktion. Zeigen Sie, dass für $x \in [X_-, X_+]$ gilt

$$\mathrm{rd}(x) = (1 + \epsilon_x)x, \quad \text{mit } |\epsilon_x| \leq \frac{1}{2}\epsilon \text{ und } \epsilon = \beta^{1-r}.$$

Hinweis: Sie können ohne Beweis voraussetzen, dass für jede Zahl $x \in \mathbb{R}$ ein $e \in \mathbb{Z}$ existiert, so dass $x = m\beta^e$, mit $m = \sum_{i=1}^{\infty} m_i \beta^{-i}$.

Aufgabe 2 (Rundung) (4 Punkte)

Das gängige Verfahren zum Runden von Zahlen ist das Aufrunden (natürliche Rundung). Bei Fließkommazahlen $\mathbb{F}(\beta, r, s)$ mit geradem β wird jedoch ein anderes Verfahren verwendet, die gerade Rundung.
Wenn x eine auf r Stellen zu rundende Zahl ist und $\mathrm{left}(x) := \max\{y \in \mathbb{F} \mid y \leq x\}$ sowie $\mathrm{right}(x) := \min\{y \in \mathbb{F} \mid y \geq x\}$ dann gilt beim Aufrunden:

$$rd(x) = \begin{cases} \mathrm{left}(x) & \text{falls} \quad 0 \leq m_{r+1} < \beta/2 \\ \mathrm{right}(x) & \text{falls} \quad \beta/2 \leq m_{r+1} < \beta \end{cases}$$

Beim geraden Runden ist dagegen:

$$rd(x) = \begin{cases} \mathrm{left}(x) & \text{falls} \quad |x - \mathrm{left}(x)| < |x - \mathrm{right}(x)| \text{ oder} \\ & \quad (|x - \mathrm{left}(x)| = |x - \mathrm{right}(x)| \text{ und } m_r \text{ gerade}) \\ \mathrm{right}(x) & \text{sonst} \end{cases}$$

Dabei ist m_i jeweils die i-te Stelle von x.

(a) Berechnen Sie die Folge von Fließkommazahlen

$$x_0 = x, \quad x_1 = (x_0 \ominus y) \oplus y, \quad \ldots, \quad x_n = (x_{n-1} \ominus y) \oplus y,$$

mit $x = 1.56$ und $y = -0.555$. Dabei seinen x, x_i und y Fließkommazahlen in der Darstellung $\mathbb{F}(10, 3, 1)$. Welche Ergebnisse erhalten Sie für die ersten 10 Folgenglieder mit Aufrunden bzw. mit gerader Rundung?

(b) Diskutieren Sie die Ergebnisse.

(c) Warum wird bei Fließkommazahlen das gerade Runden verwendet?

Definition 2 (Leapfrog-Verfahren)

Gegeben sei ein autonomes AWP 2. Ordnung

$$\begin{aligned} y'' &= f(y), \qquad \text{auf} \quad I := [t^0, T], \qquad & t^0, T \in \mathbb{R}_0^+, \\ y(t^0) &= y^0, & y^0 \in \mathbb{R}^n, \\ y'(t^0) &= v^0, & v^0 \in \mathbb{R}^n, \end{aligned}$$

mit $f : \mathbb{R}^n \to \mathbb{R}^n$ stetig, $n \in \mathbb{N}$. Auf I sei $I_{\Delta t} := \{t^k = t^0 + k\Delta t \mid k = 0, 1, 2, \ldots \ \wedge \ t^k \leq T\}$ ein gewähltes Gitter mit Schrittweite $\Delta t \in \mathbb{R}^+$. Dann liefert die Iterationsvorschrift

$$\begin{aligned} y^{k+1} &= y^k + \Delta t v^k + \tfrac{\Delta t^2}{2} f(y^k), \\ v^{k+1} &= v^k + \tfrac{\Delta t}{2}\Big(f(y^k) + f(y^{k+1})\Big), \quad k = 0, 1, 2, \ldots \end{aligned}$$

Approximationen $y^k \approx y(t^k)$, $v^k \approx y'(t^k)$ der Lösung des AWPs.

Aufgabe 3 (Konsistenz und Zeitinvertierbarkeit des Leapfrog-Verfahrens) (3 Punkte)

(a) Zeigen Sie, dass das Leapfrog-Verfahren konsistent ist mit Konsistenzordnung 2, d.h. dass sich die *lokalen Diskretisierungsfehler* in y und v verhalten wie $\Theta(\Delta t^2)$:

$$\begin{aligned} T_{y,\Delta t}(t^{k+1}) &:= \tfrac{1}{\Delta t}\Big(y(t^{k+1}) - y(t^k)\Big) - v(t^k) - \tfrac{\Delta t}{2} f\big(y(t^k)\big), \\ T_{v,\Delta t}(t^{k+1}) &:= \tfrac{1}{\Delta t}\Big(v(t^{k+1}) - v(t^k)\Big) - \tfrac{1}{2}\Big(f\big(y(t^k)\big) + f\big(y(t^{k+1})\big)\Big). \end{aligned}$$

(b) Zeigen Sie, dass das Leapfrog-Verfahren zeitinvertierbar ist, d.h. dass die Iterationsvorschrift mit den Zeitschrittweiten $+\Delta t$ und $-\Delta t$ angewandt auf das Paar (y^k, v^k) wieder in den Ausgangspaar resultiert. Welche Bedeutung hat die Zeitinvertierbarkeit für die Energieerhaltung?

Aufgabe 4 (n-Körper Problem) (6 Punkte)

Betrachten Sie das n-Körper Problem aus der Vorlesung. Durch Diskretisierung mit dem in Definition 2 formulierten Leapfrog-Verfahren erhalten wir ein Computermodell. Implementieren Sie dieses in C++ in der Variante mit dem n^2-Algorithmus aus der Vorlesung.

- Auf der Vorlesungshomepage finden Sie ein Code-Gerüst, dass von Ihnen nur noch um die Implementierung des Leapfrog-Verfahrens und der Beschleunigungsberechnung erweitert werden muss. Nähere Informationen dazu finden Sie in der `Readme.md`. Die Daten werden im VTk Dateiformat ausgegeben und können mit dem Programm `Paraview` visualisiert werden, welches unter `http://www.paraview.org` zum Download verfügbar ist.

- Es gibt verschiedene Varianten die Daten für die n Körper als einen zusammenhängenden Block im Speicher abzulegen. Im Großen und Ganzen gibt es die folgenden beiden Möglichkeiten, die auch beide von den Ihnen zur Verfügung gestellten Klassen unterstützt werden:

```
using Vector3D = std::array<double,3>;

// Variante 1:
struct Body {
  Vector3D r_i;
  Vector3D v_i;
  double m_i;
};

std::vector<Body> data;
```

```
// Variante 2:
struct Data {
  std::vector<Vector3D> r;
  std::vector<Vector3D> v;
  std::vector<double> m;
};

Data data;
```

  Entscheiden Sie sich für eine der beiden Möglichkeitenund begründen Sie Ihre Wahl.

- Nutzen Sie bei der Berechnung der Beschleunigungen $a_i := \sum_{j \neq i} \frac{1}{m_i} F_{ij}$ mit

$$F_{ij} := G \frac{m_i m_j}{\|r_j - r_i\|^2} \frac{r_j - r_i}{\|r_j - r_i\|}$$

  zur Steigerung der Performance die Symmetrie $F_{ij} = -F_{ji}$ aus und weichen Sie die Norm im Nenner auf, um einer Division durch null vorzubeugen.

- Messen Sie in Ihrem Code die für die Berechnungen benötigte Zeit und zeigen Sie graphisch eine $\mathcal{O}(N^2)$ Komlpexitiät. Zur Darstellung eignet sich unter anderem das Programm `gnuplot`.