Einführung in die Numerik (Potschka)

Robin Heinemann

24. April 2017

Inhaltsverzeichnis

0 Einführung

Beispiel 0.1 Simulation einer Pendelbewegung
Modellannahmen:

- Masse m an Stange
- keine Reibung
- Stange: Gewicht 0, starr, Länge l
- Auslenkung ϕ

Erste Fehlerquelle: Modellierungsfehler
Modellgleichungen:

$$F_T(\phi) = -m \cdot g \sin \phi$$

Konsistenzcheck:

$$F_T(0) = 0 \qquad \text{(Ruhelage)}$$
$$F_T\left(\frac{\pi}{2}\right) = F_G = -mg$$

Bewegungsgleichungen:

- Weg $s(t)$
- $\frac{\mathrm{d}s}{\mathrm{d}t} =: v(t)$ Geschwindigkeit

- $\frac{\mathrm{d}v}{\mathrm{d}t} =: a(t)$ Beschleunigung

Beziehungen:

- Bogenlänge $s(t) = l\phi(t)$
- 2. Newton'sches Gesetz ($F = ma$)

$$-mg\sin\phi(t) = m\frac{\mathrm{d}}{\mathrm{d}t}v(t) = m\frac{\mathrm{d}^2}{\mathrm{d}t^2}s(t) = ml\frac{\mathrm{d}^2}{\mathrm{d}t^2}\phi(t)$$

$\Rightarrow$ DGL 2. Ordnung

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\phi(t) = -\frac{g}{l}\sin\phi(t) \quad t \geq 0$$

Für eindeutige Lösung braucht man zwei Anfangsbedingungen:

$$\phi(0) = \phi_0 \quad \frac{\mathrm{d}}{\mathrm{d}t}\phi(0) = u_0$$

Lösung bei kleiner Auslenkung: Linearisiere um $\phi = 0$

$$\sin\phi = \phi - \frac{1}{3!}\phi^3 + \cdots \approx \phi$$

$$\Rightarrow \frac{\mathrm{d}^2}{\mathrm{d}t^2}\phi(t) = -\frac{g}{l}\phi(t)$$

Für $u_0 = 0$ findet man mit dem Ansatz $\phi(t) = A\cos(\omega t)$:

$$-\omega^2 A\cos(\omega t) = -\frac{g}{l}A\cos(\omega t)$$

die Lösung:

$$\phi(t) = \phi_0\cos\left(\sqrt{\frac{g}{l}}t\right)$$

Fehlerquelle: Abschneidefehler.
Numerische Lösung:
Setze $u(t) := \frac{\mathrm{d}}{\mathrm{d}t}\phi(t)$

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\phi \\ u\end{pmatrix} = \begin{pmatrix}u \\ -\frac{g}{l}\sin(\phi)\end{pmatrix}$$

Appraximation mit Differenkenquotient

$$\begin{pmatrix}u(t) \\ -\frac{g}{l}\sin\phi(t)\end{pmatrix} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\phi \\ u\end{pmatrix} = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\begin{pmatrix}\phi(t+\Delta t) - \phi(t) \\ u(t+\Delta t) - u(t)\end{pmatrix} \approx \frac{1}{\underset{\substack{\downarrow \\ >0,\ \text{klein}}}{\Delta t}}\begin{pmatrix}\phi(t+\Delta t) - \phi(t) \\ u(t+\Delta t) - u(t)\end{pmatrix}$$

Fehlerquelle: Diskretisierungsfehler
Auf Gitter $t_n = n\Delta t$ mit Werten $\phi_n = \phi(n\Delta t), u_n = u(n\Delta t)$:

$$\phi_{n+1} = \phi_n + \Delta t u_n, u_{n+1} = u_n - \Delta t\frac{g}{l}\phi_n$$

Kleinerer Diskretisierungsfehler mit zentralen Differenzen:

$$-\frac{g}{l}\sin\phi(t) = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\phi(t) \approx \frac{\phi(t+\Delta t) - 2\phi(t) + \phi(t-\Delta t)}{\Delta t^2}$$

Rukursionsformel:

$$\phi_{n+1} = 2\phi_n - \phi_{n-1} - \Delta t^2 \frac{g}{l}\sin\phi_n, n \geq 1$$

mit $\phi_1 = \phi_0 + \Delta t n_0$ (Expliziter Euler)
Letzte Fehlerquelle: Rundungsfehler

1 Fehleranalyse

1.1 Zahldarstellung und Rundungsfehler

Anforderung: Rechnen mit reellen Zahlin auf dem Computer.
Problem: Speicher endlich ($\Rightarrow$ endliche Genauigkeit).
Lösung: Gleitkommazahlen, ein **Kompromiss** zwischen:

- Umfang darstellbarer Zahlen
- Genauigkeit
- Geschwindigkeit einfacher Rechenoperationen (+, -, $\cdot$, /)

Alternativen:

- Fixkommazahlen
- logarithmische Zahlen
- Rationalzahlen

Definition 1.1 Eine (normalisierte) Gleitkommazahl zur Basis $b \in \mathbb{N}, b \geq 2$, ist eine Zahl $x \in \mathbb{R}$ der Form

$$x = \pm m \cdot b^{\pm e}$$

mit der Mantisse $m = m_1 b^{-1} + m_2 b^{-2} + \cdots \in \mathbb{R}$ und dem Exponenten $e = e_{s-1}b^{s-1} + \cdots + e_0 b^0 \in \mathbb{N}$, wobei $m_i, e_i \in \{0, \dots, b{-}1\}$. Für $x \neq 0$ ist die Darstellung durch die Normieungsvorschrift $m \neq 0$ eindeutig. Für $x = 0$ setzt man $m = 0$.

Beispiel 1.2 ($b = 10$)

- m_i: i -te Nachkommastelle der Mantisse
- e: Verschiebt das Komma um e Stellen.

$$0.314 \times 10^1 = 3.14$$
$$0.123 \times 10^6 = 123\,000$$

Auf dem Rechner stehen nur endlich viele Stellen zur Verfügung:
r Ziffern + 1 Vorzeichen für Mantisse m
s Ziffern + 1 Vorzeichen für Exponenten.
Für $x = \pm[m_1 b^{-1} + \dots + m_r b^{-r}] \cdot b^{\pm[e_{s-1} b^{s-1} + \dots + e_0 b^0]}$ muss man alsu nur $(\pm)[m_1 \dots m_r](\pm)[e_{s-1} \dots e_0]$ ebspeichern. Wählt man $b = 2$, so gilt $m_i, e_i \in \{0, 1\}$ und x kann mit $2 + r + s$ Bits gespeichert werden (Maschinenzahlen). Maschinenzahlen bilden das numerische Gleitkommagitter $A = A(b, r, s)$

Beispiel 1.3 **($b = 2, r = 3, s = 1$)**

$$\begin{aligned} m &= \frac{1}{2} + m_2 \frac{1}{4} + m_3 \frac{1}{8} \in \{\frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}\} \\ e &= e_0 \in \{0, 1\} \end{aligned}$$

Da A endlich ist, gibt es eine größte/kleinste darstellbare Zahl:

$$\begin{aligned} x_{\{}min/max\} &= \pm(b-1)[b^{-1} + \dots + b^{-r}] \cdot b^{(b-1)[b^{s-1} + \dots + b^0]} \\ &= \pm(1 - b^{-r}) \cdot b^{(b^s - 1)} \end{aligned}$$

sowie eine kleinste positive/größte negative Zahl:

$$\begin{aligned} x_{posmin/negmax} &= \pm b^{-1} \cdot b^{-(b-1)[b^{s-1} + \dots + b^0]} \\ &= b^{-b^s} \end{aligned}$$

Das gängigste Format ist das IEEE-Format, das auch hinter dem Python-Datentyp float steht:

$$x = \pm m \cdot 2^{c-1022}$$

Dieser Datentyp ist 64 Bit (8 Byte) groß (dopplte Genauigkeit, double). Davon speichert 1 Bit das Vorzeichen, 52 Bits die Mantisse $m = 2^{-1} + m_2 2^{-2} + \dots + m_{53} 2^{-53}$ und 11 Bits die Charakteristik $c = c_0 2^0 + \dots + c_{10} 2^{10}$, mit $m_i, c_i \in \{0, 1\}$. Es gibt folgende spezielle Werte:

- Alle $c_i, m_i = 0 : x = \pm 0$
- Alle $m_i = 0, c_i = 1 : x = \pm\infty$
- Ein $m_i \neq 0$, alle $c_i = 1$: x = NaN (not a number)

Für c bleibt damit ein Bereich von $\{0, \dots, 2046\}$ beziehungsweise $c - 1022 \in \{-1022, \dots, 1024\}$. Damit gilt:

- $x_{max} \approx 2^{1024} \approx 1.8 \times 10^{308}, x_{min} = -x_{max}$
- $x_{posmin} = 2^{-1022} \approx 2.2 \times 10^{-308}, x_{negmax} = -x_{posmin}$

Ausgangsdaten $x \in \mathbb{R}$ einer numerischen Aufgabe und die Zwischenergebnisse einer Rechnung müssen durch Maschinenzahlen dargestellt werden. Für Zahlen des „zulässigen“ Bereichs $D = [x_{min}, x_{negmax}] \cup \{0\} [x_{posmin}, x_{max}]$ wird eine Rundungsoperation $\mathrm{rd} : D \to A$ verwendet, die

$$|x - \mathrm{rd}\, x| = \min_{y \in A} |x - y| \forall x \in D$$

erfüllt.

Beispiel 1.4 (Natürliche Rundund im IEEE-Format)

$$\mathrm{rd}(x) = \mathrm{sgn}(x) \cdot \begin{cases} 0, m_1, \dots, m_{53} \cdot 2^e & m_{54} = 0 \\ (0, m_1, \dots, m_{53} + 2^{-53}) \cdot 2^e & m_{54} = 1 \end{cases}$$

Rundungsfehler:

- absolut:
$$|x - \mathrm{rd}(x)| \leq \frac{1}{2} b^{-r} b^e$$
- relativ:
$$\left| \frac{x - \mathrm{rd}(x)}{x} \right| \leq \frac{1}{2} \frac{b^{-r} b^e}{|m| b^e} \leq \frac{1}{2} b^{-r+1}$$

Der relative Fehler ist für $x \in D \quad \{0\}$ beschränkt durch die „Maschienengenauigkeit“

$$eps = \frac{1}{2} b^{-r+1}$$

Für $x \in D$ ist $\mathrm{rd}(x) = x(1+\varepsilon), |(|\varepsilon) \leq eps$. Für das IEEE-Format (double)

$$eps = \frac{1}{2} 2^{-52} \approx 10^{-16}$$

Arithmetische Grundoperationen

$$* : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, * \in \{x, -, +, /\}$$

werden auf dem Rechner ersetzt durch Maschinenoperationen:

$$\circledast : A \times A \to A$$

Dies ist normalerweise für $x, y \in A$ und $x * y \in D$ realisiert durch

$$x \circledast y := \mathrm{rd}(x * y) = (x * y)(1 + \varepsilon), |\varepsilon| \leq eps$$

Dazu werden die Operationen maschinenintern (unter Verwendung einer längeren Mantisse) ausgeführt, normalisiet und dann gerundet. Im Fall $x*y$ $6\in$ D gibt es eine Fehlermeldung (overflow, underflow) oder das Ergebnis NaN. Achtung: Das Assoziativ- und Distributivgesetz gilt dann nur näherungsweise. Im Allgemeinen ist für $x, y, z \in A$

$$(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$$
$$(x \oplus y) \odot z \neq (x \odot z) \oplus (y \odot z)$$

Insbesondere gilt für $|y| \leq \frac{|x|}{b} eps$

$$x \oplus y = x$$

Damit ergibt sich eine alternative Charakterisierung der Maschienengenauigkeit: eps ist die kleinste positive Zahl in A, sodass $1 \oplus eps \neq 1$