Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Christian Engwer, Dipl. Math. Dipl. Inf. Sebastian Westerheide

26.04.2012

Übung zur Vorlesung

Wissenschaftliches Rechnen

SS 2012 — Blatt 2

Abgabe: 08.05.2012, 10:00 Uhr, Briefkasten 89
Code zusätzlich per e-mail an `sebastian.westerheide@uni-muenster.de`

Aufgabe 1 (Nullstellenbestimmung mit dem Newton-Verfahren) (4 Punkte)

Sei $f : \mathbb{R} \to \mathbb{R}$ eine stetig differenzierbare reelle Funktion. Das Newton-Verfahren liefert über die Fixpunktiteration

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

zu einem geeignet gewählten Startwert $x_0 \in \mathbb{R}$ eine Folge $(x_n)_{n\in\mathbb{N}}$, die gegen eine Nullstelle von f konvergiert. Genauer kann man zeigen, dass es zu jeder Nullstelle $x \in \mathbb{R}$ eine Umgebung D gibt, so dass das Newton-Verfahren für alle Startwerte $x_0 \in D$ gegen x konvergiert, falls f in einer Umgebung von x zwei mal stetig differenzierbar ist und $f'(x)^{-1}$ existiert.

(a) Laden Sie die Dateien `function.hh`, `testfunctions.hh`, `bisection_method.hh` und `bisection_method.cc` von der Vorlesungshomepage und schauen Sie sich die darin enthaltene, leicht erweiterte Lösung von Blatt 1, Aufgabe 1 an. Was hat sich an der Prozedur `bisection_method (...)` verändert?

(b) Implementieren Sie das Newton-Verfahren in C++ auf analoge Art und Weise.

- Benutzen Sie das Interface `Function` für Funktionen $f : \mathbb{R} \to \mathbb{R}$
- Schreiben Sie eine Prozedur
```
double newton_method (const Function& f, double x_0, double eps,
                      double h, unsigned int& n_count);
```
- Die Fixpunktiteration (1) soll so lange durchgeführt werden, bis die Änderungsrate $|x_{n+1} - x_n|$ eine zu übergebende Fehlerschranke `eps` unterschreitet
- Benutzen Sie zur Approximation der Ableitung den zentralen Differenzenquotienten mit einer zu übergebende Schrittweite `h`:

$$f'(x) \approx D^c f(x) := \frac{f(x+h) - f(x-h)}{2h}$$

- Geben Sie im Referenzparameter `n_count` die benötigte Anzahl an Iterationen zurück

(c) Testen Sie Ihre Implementierung mit `eps` $= 10^{-4}$ und `h` $= 10^{-2}$ an den folgenden, in `testfunctions.hh` implementierten Funktionen:

- $f_1(x) = x^2 - 1$ (Startw.: $x_0 = 1.2$) (Nullst.: $x = 1$)
- $f_2(x) = \cos(x) - x$ (Startw.: $x_0 = 0.0$) (Nullst.: $x = 0.739085133215$)
- $f_3(x) = \sin(2x) \cdot x$ (Startw.: $x_0 = 6.3$) (Nullst.: $x = 6.283185307180$)
- $f_4(x) = x^2 - 2$ (Startw.: $x_0 = -0.1$) (Nullst.: $x = -1.414213562373$)

Schreiben Sie ein Programm, das Instanzen der Funktionen erzeugt, jeweils das Newton-Verfahren aufruft und die benötigte Anzahl an Iterationen, die ermittelten Nullstellen sowie zugehörige Funktionswerte auf der Konsole ausgibt.

(d) Schreiben Sie ein Programm, welches die in Abhängigkeit von `eps` benötigte Anzahl an Iterationen sowohl für den Bisektionsalgorithmus als auch für das Newton-Verfahren ausgibt und plotten Sie das Ergebnis mit `gnuplot`:

- Benutzen Sie die Funktion f_2, das Intervall $[0, 1]$, den Startwert $x_0 = 0.0$ und `h` $= 10^{-2}$
- Wählen Sie verschiedene Fehlerschranken `eps` $= 2^{-i}$, $i = 0, 1, 2, ...$
- Geben Sie für jedes `eps` eine Zeile folgenden Formates auf der Konsole aus

```
eps ␣ n_count_bisection ␣ n_count_newton
```

- Leiten Sie die Ausgabe des Programms in eine Datei `compare.data` um
- Laden Sie sich die Datei `compare.plt` von der Vorlesungshomepage, starten Sie `gnuplot` und führen Sie den Befehl `load 'compare.plt'` aus

Erklären Sie die einzelnen Anweisungen in der Datei `compare.plt`. Interpretieren Sie das Ergebnis, das durch den Plot darstellt wird.

Aufgabe 2 (Zahlendarstellung) (3 Punkte)

Sie haben in der Vorlesung die allgemeine Darstellung von Fließkommazahlen als $\mathbb{F}(\beta, r, s)$ kennengelernt. Dabei ist β die Basis, r die Anzahl Stellen der Mantisse und s die Anzahl Stellen des Exponenten.

(a) Bilden Sie $x_0 = \frac{1}{11}$ nach $\mathbb{F}(2, 10, 3)$ ab, d.h. geben Sie $\mathrm{rd}\left(\frac{1}{11}\right)$ mit $\mathrm{rd}: \mathbb{R} \to \mathbb{F}(2, 10, 3)$.

(b) Stellen Sie $x_1 = \frac{1}{2048}$ in $\mathbb{F}(10, 5, 1)$ dar.

Aufgabe 3 (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 $\text{left}(x) := \max\{y \in \mathbb{F} \mid y \leq x\}$ sowie $\text{right}(x) := \min\{y \in \mathbb{F} \mid y \geq x\}$ dann gilt beim Aufrunden:

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

Beim geraden Runden ist dagegen:

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

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?

Aufgabe 4 (Leapfrog) (5 Punkte)

In der Vorlesung haben Sie das Leapfrog-Verfahren

$$\begin{aligned} y_{k+1} &= y_k + v_{k+1/2}\,\Delta t, \qquad & k = 0, 1, 2, ..., \\ v_{k+3/2} &= v_{k+1/2} + a(y_{k+1})\,\Delta t, \end{aligned}$$

zur numerischen Lösung folgender Klasse von Anfangswertproblemen kennengelernt:

$$\begin{aligned} y'' = v' = a(y) \quad \text{auf} \quad [0, \infty), \\ y(0) = y_0, \quad v(t_{1/2}) = v_{1/2}. \end{aligned}$$

(a) Implementieren Sie das Verfahren in C++.

(b) Zeigen Sie mittels Taylorentwicklung, dass sich der "lokale Diskretisierungsfehler"

$$T_{\Delta t}(t_{k+1}) := \frac{1}{\Delta t}\Big(y(t_{k+1}) - y(t_k)\Big) - v(t_{k+1/2})$$

verhält wie $O(\Delta t^2)$.