Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2010/11
09. Dezember 2010

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Aufgabe 8.1*. Alternativ zum Bisektionsverfahren aus der Vorlesung kann man zur Berechnung einer Nullstelle einer Funktion $f : [a, b] \to \mathbb{R}$ das *Newton-Verfahren* verwenden. Ausgehend von einem Startwert x_0 definiert man induktiv eine Folge (x_n) wie folgt: Zu gegebenem x_k sei x_{k+1} die Nullstelle der Tangente an den Graphen von f im Punkt $(x_k, f(x_k))$, d.h. $x = x_{k+1}$ erfüllt $0 = f(x_k) + f'(x_k)(x - x_k)$. Auflösen nach x zeigt

$$x_{k+1} = x_k - f(x_k)/f'(x_k).$$

Realisieren Sie das Newton-Verfahren in einer Funktion `newton(f,fprime,x0,tau)`, wobei die Iteration abgebrochen wird, falls entweder

$$|f'(x_n)| \le \tau$$

oder

$$|f(x_n)| \le \tau \quad \text{und} \quad |x_n - x_{n-1}| \le \begin{cases} \tau & \text{für } |x_n| \le \tau, \\ \tau|x_n| & \text{sonst} \end{cases}$$

gilt. In jedem Fall werde $\widetilde{z} := x_n$ als Approximation der gesuchten Nullstelle zurückgegeben, wobei im ersten Fall zusätzlich eine Warnung ausgegeben werden soll. — Neben x_n sollen die Folgen $(x_0, \dots, x_n)$ der Nullstellen und der dazugehörigen Funktionswerte zurückgegeben werden. Testen Sie Ihre Implementierung mit der Funktion $f(x) = x^2 + e^x - 2$. Speichern Sie den Source-Code unter `newton.m` in das Verzeichnis `serie08`.

Aufgabe 8.2*. Für $x > 0$ konvergiert die Folge

$$x_1 := \frac{1}{2}(1 + x), \quad x_{n+1} := \frac{1}{2}\Big(x_n + \frac{x}{x_n}\Big) \quad \text{für } n \ge 1$$

gegen $\sqrt{x}$. Schreiben Sie eine Funktion `sqrt_`, die für gegebene $x > 0$ und $\tau > 0$ als Ergebnis das erste Folgenglied $y = x_n$ zurückgibt, für das gilt

$$\frac{|x_n - x_{n+1}|}{|x_n|} \le \tau \quad \text{oder} \quad |x_n| \le \tau.$$

Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x eingelesen und neben der Approximation x_n von $\sqrt{x}$ auch der exakte Wert sowie der absolute Fehler $|x_n - \sqrt{x}|$ ausgegeben werden. Speichern Sie den Source-Code unter `sqrt.m` in das Verzeichnis `serie08`. Welcher Zusammenhang besteht mit dem Newton-Verfahren aus Aufgabe 8.1?

Aufgabe 8.3*. Für eine differenzierbare Funktion $f : [a, b] \to \mathbb{R}$ kann man die Ableitung $f'(x)$ in einem festen Punkt $x \in \mathbb{R}$ durch den einseitigen Differenzenquotienten

$$\Phi(h) := \frac{f(x + h) - f(x)}{h} \quad \text{für } h > 0$$

approximieren. Nach Taylor-Formel gilt für $f \in C^2(\mathbb{R})$ die Fehlerabschätzung $|f'(x) - \Phi(h)| = \mathcal{O}(h)$. Beweisen Sie diese Aussage mathematisch! Schreiben Sie eine Funktion `diff(f,x,h0,tau)`, die für $h_n := 2^{-n}h_0$ die Folge der $\Phi(h_n)$ berechnet, bis gilt

$$|\Phi(h_n) - \Phi(h_{n+1})| \leq \begin{cases} \tau & \text{für } |\Phi(h_n)| \leq \tau, \\ \tau\,|\Phi(h_n)| & \text{anderenfalls.} \end{cases}$$

Die Funktion liefere in diesem Fall die vollständige Folge $(\Phi(h_0), \dots, \Phi(h_n))$ der Iterierten zurück. Speichern Sie den Source-Code unter `diff.m` in das Verzeichnis `serie08`.

Aufgabe 8.4*. Es sei $L \in \mathbb{R}^{n\times n}$ eine untere Dreiecksmatrix mit $\ell_{jj} \neq 0$ für alle $j = 1, \dots, n$. Dann ist L invertierbar, und die Inverse lässt sich wie folgt rekursiv berechnen: Wir schreiben L in Block-Form

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}$$

mit $L_{11} \in \mathbb{R}^{p\times p}$, $L_{21} \in \mathbb{R}^{q\times p}$ und $L_{22} \in \mathbb{R}^{q\times q}$, wobei $n = p + q$ gilt. Man beachte, dass L_{11} und L_{22} wieder untere Dreiecksmatrizen sind mit $\ell_{jj} \neq 0$. Üblicherweise wählt man $p = n/2$, falls n gerade ist, bzw. $p = (n-1)/2$, falls n ungerade ist. Offensichtlich wird die Inverse L^{-1} dann durch

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}$$

gegeben. Schreiben Sie eine Funktion `invertL`, die die Inverse L^{-1} rekursiv berechnet. Speichern Sie den Source-Code unter `invertL.m` in das Verzeichnis `serie08`. Sie können die Korrektheit Ihrer Funktion mit Hilfe der Matlab-Funktion `inv` überprüfen.

Aufgabe 8.5. Schreiben Sie eine (rekursive) Funktion `papierschnitt`, die alle Möglichkeiten visualisiert, wie ein Papierbogen der ganzzahligen Länge `laenge` in Papierbahnen der Länge 1 und 2 geschnitten werden kann. — D.h. man stelle eine natürliche Zahl $n =$ `laenge` auf alle möglichen Weisen als Summe $n = \sum_{j=1}^{k} \sigma_j$ mit Summanden $\sigma_j \in \{1, 2\}$ dar. Dabei soll die Reihenfolge beachtet werden. Für `laenge=4` gibt es beispielsweise 5 Möglichkeiten:

- $4 = 2 + 2$
- $4 = 2 + 1 + 1$
- $4 = 1 + 2 + 1$
- $4 = 1 + 1 + 2$
- $4 = 1 + 1 + 1 + 1$

Aufgabe 8.6. Das Newton-Verfahren aus Aufgabe 8.1 benötigt neben der Funktion `f` auch eine Funktion `fprime`, die die Ableitung f' der Funktion f auswertet. Alternativ kann man $f'(x_k)$ durch den Differenzenquotienten $\Phi_h(x_k)$ aus Aufgabe 8.3 ersetzen. Realisieren Sie dieses Vorgehen indem Sie eine Funktion `newton(f,x0,h0,tau)` schreiben, die zur Approximation der Abeitung $f'(x_k)$ das Ergebnis von `diff(f,xk,h0,tau)` verwendet.

Aufgabe 8.7. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und 3 Vektoren $I, J, A \in \mathbb{R}^N$ mit $N \leq mn$, die eine schwachbesetzte Matrix in naiver Speicherung repräsentieren. Schreiben Sie eine Funktion `naive2ccs(I,J,A,m,n)`, die als Ergebnis die entsprechenden Vektoren des CCS-Formats zurückliefert.

Aufgabe 8.8. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und Vektoren $I, A \in \mathbb{R}^N$ sowie $J \in \mathbb{R}^{n+1}$, die eine schwach besetzte $(m \times n)$-Matrix in CCS-Speicherung repräsentieren. Schreiben Sie eine Funktion `ccs2naive(I,J,A,m,n)`, die als Ergebnis die entssprechenden Vektoren bei naiver Speicherung zurückgibt.

Aufgabe 8.9. Gegeben seien Dimensionen $m, n \in \mathbb{N}$ und 3 Vektoren $I, A \in \mathbb{R}^N$ und $J \in \mathbb{R}^{n+1}$, die eine schwachbesetzte $(m \times n)$-Matrix in CCS-Speicherung repräsentieren, sowie ein Vektor $x \in \mathbb{R}^n$. Schreiben Sie mittels geeigneter (möglichst weniger) Schleifen eine Matrix-Vektor-Multiplikation `mvmsparse(I,J,A,m,n,x)`.

Aufgabe 8.10. Wie viele Möglichkeiten gibt es eine 1 €-teure Wurstsemmel in Münzen zu bezahlen? Die Reihenfolge der Münzen soll dabei keine Rolle spielen. Eine Möglichkeit wäre beispielsweise: 1x 50-Cent, 2x 20-Cent, 1x 10-Cent. Schreiben Sie eine (rekursive) Funktion `wurstsemmel`, die zu gegebenem Betrag die Anzahl der Möglichkeiten zurückliefert.