Dirk Praetorius
Michele Ruggeri
Bernhard Stiftner

Sommersemester 2016
12.04.2016

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. Die Quotientenfolge $(a_{n+1}/a_n)_{n\in\mathbb{N}}$ zur Fibonacci-Folge $(a_n)_{n\in\mathbb{N}}$,

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \text{für } n \geq 2,$$

konvergiert gegen den goldenen Schnitt $(1+\sqrt{5})/2$. Insbesondere konvergiert die Differenz

$$b_n := \frac{a_{n+1}}{a_n} - \frac{a_n}{a_{n-1}}$$

gegen Null. Schreiben Sie eine *nicht-rekursive* Funktion `cauchy`, die zu gegebenem $k \in \mathbb{N}$ die kleinste Zahl $n \in \mathbb{N}$ mit $|b_n| \leq 1/k$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das die Zahl $k \in \mathbb{N}$ einliest und den zugehörigen Index $n \in \mathbb{N}$ ausgibt. Speichern Sie den Source-Code unter `goldenerSchnitt.c` in das Verzeichnis `serie05`.

Aufgabe 5.2. 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. Schreiben Sie eine Funktion `double diff(double x, double h0, double tau)`, die für $h_n := 2^{-n} h_0$ ($n \in \mathbb{N}$) die Folge der $\Phi(h_n)$ berechnet, bis gilt

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

Die Funktion liefere in diesem Fall $\Phi(h_n)$ als Approximation von $f'(x)$ zurück. Stellen Sie mittels `assert` sicher, dass $h_0 > 0$ ist. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` arbeiten. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x, h_0 und τ eingelesen werden und $\Phi(h_n)$ ausgegeben wird. Speichern Sie den Source-Code unter `diff.c` in das Verzeichnis `serie05`.

Aufgabe 5.3. Alternativ zum Bisektionsverfahren aus der Vorlesung kann eine Nullstelle von $f : [a,b] \to \mathbb{R}$ auch mit dem *Sekantenverfahren* berechnet werden. Dabei sind x_0 und x_1 gegebene Startwerte und man definiert induktiv x_{n+1} als Nullstelle der Geraden durch $(x_{n-1}, f(x_{n-1}))$ und $(x_n, f(x_n))$, d.h.

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

Schreiben Sie eine Funktion `sekante(x0,x1,tau)` die die Folge der Iterierten berechnet, bis entweder

$$|f(x_n) - f(x_{n-1})| \leq \tau$$

oder

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

gilt. Es werde dann x_n als Approximation einer Nullstelle z_0 von f zurückgegeben. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` arbeiten. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 und x_1 eingelesen werden und x_n ausgegeben wird. Speichern Sie den Source-Code unter `sekante.c` in das Verzeichnis `serie05`.

Aufgabe 5.4. Eine Variante zur Berechnung einer Nullstelle einer Funktion $f : [a, b] \to \mathbb{R}$ ist das *Newton-Verfahren*. Ausgehend von einem Startwert x_0 definiert man induktiv eine Folge $(x_n)_{n\in\mathbb{N}}$ durch

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

Man realisiere das Newton-Verfahren in einer Funktion `double newton(double x0, double 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. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` und Ableitung `double fstrich(double x)` arbeiten. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 eingelesen und x_n ausgegeben wird. Speichern Sie den Source-Code unter `newton.c` in das Verzeichnis `serie05`.

Aufgabe 5.5. Das Newton-Verfahren aus Aufgabe 5.4 benötigt neben der Funktion `f` auch eine Funktion `fstrich`, die die Ableitung f' der Funktion f auswertet. Alternativ kann man $f'(x_k)$ durch den Differenzenquotienten $\Phi_h(x_k)$ aus Aufgabe 5.2 ersetzen. Realisieren Sie dieses Vorgehen indem Sie eine Funktion `double newton2(double x0, double h0, double tau)` schreiben, die zur Approximation der Ableitung $f'(x_k)$ das Ergebnis von `diff(xk,h0,tau)` verwendet. Stellen Sie mittels `assert` sicher, dass $h_0 > 0$ ist. Speichern Sie den Source-Code unter `newton2.c` in das Verzeichnis `serie05`.

Aufgabe 5.6. Für $p \in [1, \infty)$ ist die ℓ_p-Norm auf $\mathbb{R}^n$ definiert durch

$$\|x\|_p := \Big(\sum_{j=1}^{n} |x_j|^p \Big)^{1/p}.$$

Schreiben Sie eine Funktion `pnorm`, die einen Vektor $x \in \mathbb{R}^n$, dessen Länge n sowie $p \in [1, \infty)$ übernimmt und $\|x\|_p$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x und p eingelesen werden und $\|x\|_p$ ausgegeben wird. Die Dimension $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm sein, die Funktion `pnorm` soll aber beliebige Dimension zulassen. Testen Sie Ihr Programm mit verschiedenen Werten für p bei festem Vektor x. Was beobachten Sie für $p \to \infty$? Speichern Sie den Source-Code unter `pnorm.c` in das Verzeichnis `serie05`.

Aufgabe 5.7. Man erweitere *MinSort* aus der Vorlesung um einen Parameter `type`, sodass die Funktion einen Vektor $x \in \mathbb{R}^n$ wahlweise aufsteigend (`type = 1`) oder absteigend (`type = -1`) sortiert. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $x \in \mathbb{R}^n$ und die Sortierrichtung `type` eingegeben werden und der sortierte Vektor ausgegeben wird. Die Länge n des Vektors soll eine Konstante im Hauptprogramm sein, Ihre Implementierung von *MinSort* soll aber für beliebige Länge n funktionieren. Speichern Sie den Source-Code unter `minsort.c` in das Verzeichnis `serie05`.

Aufgabe 5.8. *Bubble-Sort* ist ein ineffizienter, aber kurzer Sortier-Algorithmus: Man vergleicht aufsteigend jedes Element eines Arrays x_j mit seinem Nachfolger x_{j+1} und - falls notwendig - vertauscht die beiden. Nach dem ersten Durchlauf muß zumindest das letzte Element bereits am richtigen Platz sein. Der nächste Durchlauf muß also nur noch bis zur vorletzten Stelle gehen, usw. Wie viele geschachtelte Schleifen braucht dieses Vorgehen? Schreiben Sie eine Funktion `bubblesort`, die ein gegebenes Array $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \le x_2 \le \cdots \le x_n$, und zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x einliest und in sortierter Reihenfolge ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `bubblesort` ist für beliebige Länge n zu programmieren. Bestimmen Sie den Aufwand Ihrer Funktion. Speichern Sie den Source-Code unter `bubblesort.c` in das Verzeichnis `serie05`.