Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
24.04.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

Aufgabe 6.1. 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 (*f)(double), 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

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

Die Funktion liefere in diesem Fall $\Phi(h_n)$ als Approximation von $f'(x)$ zurück. Stellen Sie mittels `assert` sicher, dass $\tau, h_0 > 0$ gilt. 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. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `diff.c` in das Verzeichnis `serie06`.

Aufgabe 6.2. 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 `double sekante(double (*f)(double), double x0, double x1, double tau)` die die Folge der Iterierten berechnet, bis entweder

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

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. 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. Stellen Sie mittels `assert` sicher, dass $\tau > 0$ gilt. 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. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `sekante.c` in das Verzeichnis `serie06`.

Aufgabe 6.3. 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

$$x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1}) \quad \text{für } n \ge 1.$$

Man realisiere das Newton-Verfahren in einer Funktion `double newton(double (*fct)(double), double (*fctprime)(double), double x0, double tau)`, die die Approximation x_n der Nullstelle zurückgibt, wobei die Iteration abgebrochen wird, falls entweder

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

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. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist. Stellen Sie mittels `assert` sicher, dass $\tau > 0$ gilt. Die Funktion $f : [a, b] \to \mathbb{R}$ und die dazugehörige Ableitung $f' : [a, b] \to \mathbb{R}$ sollen mittels Funktionspointer an die Funktion `newton` übergeben werden. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 und τ eingelesen werden und x_n ausgegeben wird. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `newton.c` in das Verzeichnis `serie06`.

Aufgabe 6.4. Das Newton-Verfahren aus Aufgabe 6.3 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 6.1 ersetzen. Realisieren Sie dieses Vorgehen indem Sie eine Funktion `double newton2(double (*f)(double), double x0, double h0, double tau)` schreiben, die zur Approximation der Ableitung $f'(x_k)$ das Ergebnis von `diff(f, xk, h0, tau)` verwendet. Stellen Sie mittels `assert` sicher, dass $\tau, h_0 > 0$ gilt. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `newton2.c` in das Verzeichnis `serie06`.

Aufgabe 6.5. Schreiben Sie eine Funktion, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das Minimum $\min_{j=1}^n x_j$, das Maximum $\max_{j=1}^n x_j$ und den Mittelwert $\frac{1}{n}\sum_{j=1}^n x_j$ berechnet und zurückgibt. Schreiben Sie zwei Versionen der Funktion mit folgenden Signaturen:

```
double* minmaxmean1(double* x,int n);
void minmaxmean2(double* x,int n,double* min,double* max,double* mean)
```

Ferner schreibe man ein aufrufendes Hauptprogramm, das n und x einliest und die von `minmaxmean1` und `minmaxmean2` berechneten Kenngrößen ausgibt. Speichern Sie den Source-Code unter `minmaxmean.c` in das Verzeichnis `serie06`.

Aufgabe 6.6. Schreiben Sie eine Funktion `void unique(double* x, int* n)`, die einen Vektor $x \in \mathbb{R}^n$ aufsteigend sortiert, doppelte Einträge streicht und den Vektor in gekürzter Form zurückgibt. Die Funktion soll also beispielsweise den Vektor $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ durch den Vektor $x = (1, 3, 4, 5) \in \mathbb{R}^4$ überschreiben. Der Vektor x soll mit dem gekürzten Vektor überschrieben werden d.h. Vektorlänge und dynamisches Koeffizientenfeld müssen ggf. angepasst werden! Schreiben Sie ein aufrufendes Hauptprogramm, in dem die Vektorlänge n und der Vektor $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird. Testen Sie Ihre Implementierung mit passenden Beispielen! Speichern Sie den Source-Code unter `unique.c` in das Verzeichnis `serie06`.

Aufgabe 6.7. Wo liegen die Fehler im folgenden Programm?

```
#include <stdio.h>

void square(double* x)
{
  double* y;
  x=(*y)*(*x);
}

int main(){
  double x=2.1;
  square(&x);
  printf("x^2=%f\n",x);
  return 0;
}
```

Verändern Sie *nur* die Funktion `square`, so dass der Output des Codes den Erwartungen entspricht.

Aufgabe 6.8. Was ist der Unterschied und der Zusammenhang zwischen einer Variable und einem Pointer? Was könnten Vor- und Nachteile dieser Konstrukte sein?
Schreiben Sie eine Funktion `swap`, welche die Werte zweier Variablen `x` und `y` vertauscht. Warum funktioniert das folgende Vorgehen nicht?

```
void swap(double x, double y)
{
   double tmp;
   tmp = x;
   x = y;
   y = tmp;
}
```

Speichern Sie den Source-Code unter `swap.c` in das Verzeichnis `serie06`.