# Algorithmische Mathematik I

Wintersemester 2017/18
Prof. Dr. Ira Neitzel
AR. Dr. Tino Ullrich

universitätbonn

## Übungsblatt 6.

Abgabe am **20.11.2017 vor der Vorlesung**.

**Aufgabe 1.** (Konvergenzordnung)

Aus der Vorlesung kennen Sie die Definition der *Konvergenzordnung*: Sei $(x_n)_{n\in\mathbb{N}}$ eine kovergente Folge reeller Zahlen mit $x_n \xrightarrow[n\to\infty]{} x^*$.

- Existiert eine Konstante $0 < c < 1$ mit

  $$|x_{n+1} - x^*| \leq c|x_n - x^*| \quad , \quad n > n_0,$$

  so hat die Folge mindestens Konvergenzordnung 1.

- Gilt $|x_{n+1} - x^*| \leq c_n|x_n - x^*|$ mit $c_n \to 0$ für $n \to \infty$, so spricht man von *superlinearer Konvergenz*.

- Sei $p > 1$. Existiert eine Konstante $c > 0$, so dass

  $$|x_{n+1} - x^*| \leq c|x_n - x^*|^p\,,$$

  so hat die Folge mindestens Konvergenzordnung $p$.

  Geben Sie für diese Folgen die Konvergenzordnung an.

  a. $x_n := (1+\varepsilon)^{-n}$ mit festem $\varepsilon > 0$,
  b. $x_n := 2^{-2^{n/2}}$,
  c. $x_n := 1/n^2$,
  d. $x_{n+1} := \varphi(x_n)$ mit $x_0 = 1/3$ und $\varphi(x) = x/\ln(1/x)$,
  e. $x_{n+1} := 2 + (x_n - 1)^{1/(2x_n-4)}(x_n - 2)^2$ mit $x_0 = 5/2$.

(6 Punkte)

**Aufgabe 2.** (Konvergenz des Bisektionsverfahrens)

Gegeben sei eine stetige Funktion $f : [a, b] \to \mathbb{R}$ mit $f(a)f(b) < 0$, die eine eindeutige Nullstelle $x^\star \in (a, b)$ besitzt. Zeigen Sie, dass die $k$-te Iterierte $x^{(k)}$ des Bisektionsverfahrens der Abschätzung

$$|x^{(k)} - x^\star| < \frac{1}{2^k}(b - a)$$

genügt. Mit welcher Konvergenzordnung (siehe Aufgabe 1) konvergiert die Folge der Intervalllängen gegen 0? Was kann man über die Konvergenzordnung der Folge $x^{(k)}$ aussagen?

(6 Punkte)

**Aufgabe 3.** (Asymptotische Laufzeitnotationen)

Zur Geschwindigkeits- und Speicherverbrauchsanalyse bei Algorithmen verwendet man die Notationen ($g : \mathbb{N} \to (0, \infty)$)

$$\mathcal{O}(g(n)) := \{f \colon \mathbb{N} \to (0,\infty) \,:\, \exists c > 0 \text{ und } \exists n_0 \in \mathbb{N} \text{ mit } f(n) \leq c \cdot g(n)\ \forall n \geq n_0\},$$
$$o(g(n)) := \{f \colon \mathbb{N} \to (0,\infty) \,:\, \forall c > 0\, \exists n_0 \in \mathbb{N} \text{ mit } f(n) \leq c \cdot g(n)\ \forall n \geq n_0\}\,.$$

Die Funktion $f(n)$ kann beispielsweise die Anzahl der Rechenschritte eines Programms in Abhängigkeit der Eingabegröße $n$ angeben.

a. Definieren Sie $\mathcal{O}(g(n)$ und $o(g(n))$ über die Folge $(f(n)/g(n))_{n\in\mathbb{N}}$.

b. Beweisen oder wiederlegen Sie die folgenden Aussagen:

- $n^6 \in \mathcal{O}(2^n)\,,$
- $42n^3 + 13n^2 + 2n + 500 \in \mathcal{O}(n^3)\,,$
- $42n^3 + 13n^2 + 2n + 500 \in o(n^2)\,.$

c. Sortieren Sie die folgenden Funktionen bezüglich ihrer "Größenordnung", d.h. geben Sie eine Reihenfolge $g_1, \ldots, g_k$ an mit $g_i \in \mathcal{O}(g_{i+1})$. Geben Sie dabei an, wenn mehrere $g_i$ dieselbe Größenordnung haben.

$$(\sqrt{2})^{\log_2(n)}, \quad n^2, \quad \left(\frac{3}{2}\right)^n, \quad n^3, \quad 2^n, \quad \log_2^2(n), \quad 2^{\log_2(n)}, \quad \sqrt{n},$$

$$10^4, \quad e^n, \quad n\log_2(n), \quad n!, \quad 2^{(2^n)}$$

**Bemerkung:** Aufgabenteil c) dient dazu ein Gefühl für Komplexitäten zu erhalten. Die Einordnung der letzten beiden Terme $n!, 2^{(2^n)}$ darf ohne Begründung angegeben werden und diese beiden Terme zählen nicht in die Bewertung. Zudem darf in der gesamten Aufgabe vorausgesetzt werden, dass z.B. $1/n^k$ oder $n^k/e^n$ eine Nullfolge ist.

(8 Punkte)

**Programmieraufgabe 1.** (Elementare Vektoroperationen)

Implementieren Sie für die nachfolgenden Aufgaben in C/C++ geeignete Funktionen der Form

```
void PROGNAME(var1,var2,...)
```

z.B.

```
void PROGNAME(double *res, double *x, int n)
{
    // Programm-Code der Routine,
    // z.B. Berechnung des komponentenweisen Vorzeichens
    // eines Vektors x der Laenge n.
    int i;
    for (i=0 ; i<n ; ++i)
        {if (x[i]<0)
            res[i] = -1;
         else
            res[i] = 1;
         }
}
```

Gegeben seien Vektoren $x, y, z \in \mathbb{R}^n$ sowie die reelle Zahl $\alpha \in \mathbb{R}$. Implementieren Sie

a. das komponentenweise Maximum `VDMAX(z,x,y,n)`

$$z_i = \max\{x_i, y_i\} \qquad \text{für } i = 1, \ldots, n,$$

b. die komponentenweisen Addition `PLUS(z,x,y,n)`

$$z_i = x_i + y_i \qquad \text{für } i = 1, \ldots, n,$$

c. die komponentenweise Multiplikation `MULTVEKTOR(z,x,y,n)`

$$z_i = x_i \cdot y_i \qquad \text{für } i = 1, \ldots, n,$$

d. die Multiplikation `MULTZAHL(z,x,`$\alpha$`,n)` des Vektors $x$ mit der Zahl $\alpha$

$$z_i = \alpha x_i \qquad \text{für } i = 1, \ldots, n,$$

e. das euklidischen Skalarprodukt `SCALPR(`$\beta$`,x,y,n)` mit dem Ergebnis $\beta \in \mathbb{R}$

$$\beta = \sum_{i=1}^{n} x_i y_i,$$

f. die komponentenweise Division `DIVVEKTOR(z,x,y,n)`

$$z_i = \begin{cases} x_i/y_i, & \text{falls } y_i \neq 0, \\ 0, & \text{falls } y_i = 0 \end{cases} \qquad \text{für } i = 1, \ldots, n,$$

g. die euklidische Norm (Länge) eines Vektors `NORM(`$\ell$`,x,n)`

$$\ell = \Big( \sum_{i=1}^{n} |x_i|^2 \Big)^{1/2}.$$

Nutzen Sie als Datenstruktur *dynamische arrays*. Informieren Sie sich dabei über die $C-$Funktionen `malloc()` und `free()`. Es darf aber auch mit den C++ Operatoren `new` und `delete` gearbeitet werden.

Testen Sie ihre Funktionen anschließend, indem Sie ein C/C++-Programm schreiben, dass zunächst eine Länge $n$ (Dimension) einliest. Anschließend werden Zufallsvektoren aus $[0,1]^n$ erzeugt und eine Zahl $\alpha \in (0,1)$ eingelesen und die Operationen a)-g) durchführt. Bennutzen Sie die Funktionen `rand()` und `srand()` aus der C-Standardbibliothek.

Lesen Sie außerdem eine Zahl "Anzahl der Versuche" ein und starten Sie folgendes (Zufalls-)Experiment. Bestimmen Sie die relative Häufigkeit, dass $m$ unabhängig gezogene Zufallsvektoren aus $[0,1]^n$ eine Länge (Norm) kleiner oder gleich 1 haben? Was beobachten Sie mit wachsender Dimension $n$?

(10 Punkte)

Die Programmieraufgabe wird in der Woche vom 20.11.2017 bepunktet

**Die Fachschaft Mathematik feiert am 23.11. ihre Matheparty in der N8schicht! Der VVK findet am Mo. 20.11., Di. 21.11. und Mi 22.11. in der Mensa Poppelsdorf statt. Weitere Infos gibt es auch auf fsmath.uni-bonn.de**