Hochschule Aalen
Fakultät Elektronik und Informatik
Studiengang Informatik

Analysis und Lineare Algebra

Vorlesung im Wintersemester 2014/2015
Prof. Dr. habil. Christian Heinlein

6. Übungsblatt (10. November 2014)

Aufgabe 10: Potenzgesetze

Beweisen Sie die nachfolgenden Rechenregeln für $a > 0$ und $x, y \in \mathbb{R}$ unter Verwendung der folgenden Definitionen und Regeln (vgl. Kapitel 2, Definition 4 und Beispiel 8):
$e^x := E(x)$, $\ln x$ ist die Umkehrfunktion von e^x, $a^x := e^{x \ln a}$, $E(0) = 1$, $E(x)\,E(-x) = 1$

a) $a^0 = 1$

✎

$a^0 = e^{0 \cdot \ln a} = e^0 = E(0) = 1$

✐

b) $a^1 = a$

✎

$a^1 = e^{1 \cdot \ln a} = e^{\ln a} = a$

✐

c) $a^{-x} = \frac{1}{a^x}$

✎

$a^{-x} = e^{-x \ln a} = E(-x \ln a) = \frac{1}{E(x \ln a)} = \frac{1}{e^{x \ln a}} = \frac{1}{a^x}$

✐

d) $\left(a^x\right)^y = a^{xy}$

✎

$\left(a^x\right)^y = \left(e^{x \ln a}\right)^y = e^{y \ln\left(e^{x \ln a}\right)} = e^{y(x \ln a)} = e^{(xy)\ln a} = a^{xy}$

✐

Aufgabe 11: Taylorpolynome

a) Wie lauten die Ableitungen $f^{(k)}(x)$ der Funktion $f(x) = \cos x$?
Wie lauten ihre Werte $f^{(k)}(0)$ an der Stelle 0?
Wie lautet die Taylorreihe der Funktion?

✎

- $f'(x) = -\sin x,\ f''(x) = -\cos x,\ f^{(3)}(x) = \sin x,\ f^{(4)}(x) = \cos x = f(x)$
- $f^{(2k)}(x) = (-1)^k \cos x$, d. h. $f^{(2k)}(0) = (-1)^k$ für $k = 0, 1, 2, \ldots$

- $f^{(2k+1)}(x) = (-1)^{k+1} \sin x$, d. h. $f^{(2k+1)}(0) = 0$ für $k = 0, 1, 2, \ldots$
- $f(x) = \sum\limits_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} x^k = \sum\limits_{k=0}^{\infty} \frac{f^{(2k)}(0)}{(2k)!} x^{2k} + \sum\limits_{k=0}^{\infty} \frac{f^{(2k+1)}(0)}{(2k+1)!} x^{2k+1} = \sum\limits_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!} = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} \pm \ldots$

✐

b) Gegeben sei die Funktion $f(x) = \ln(1 + x)$ für $x > -1$.

(i) Zeigen Sie durch vollständige Induktion: $f^{(k)}(x) = (-1)^{k-1} \frac{(k-1)!}{(1+x)^k}$ für $k = 1, 2, \ldots$

✎

Induktionsanfang $k = 1$:

$f^{(1)}(x) = f'(x) = \frac{1}{1+x} = (-1)^{1-1} \frac{(1-1)!}{(1+x)^1}$ = Behauptung für $k = 1$

Induktionsschritt $k \to k + 1$:

$f^{(k+1)}(x) = \left(f^{(k)}(x)\right)' = \left((-1)^{k-1} \frac{(k-1)!}{(1+x)^k}\right)' = (-1)^{k-1} (k-1)! \frac{-k}{(1+x)^{k+1}} = (-1)^k \frac{k!}{(1+x)^{k+1}} =$
Behauptung für $k + 1$

✐

(ii) Geben Sie für die Funktion f das Taylorpolynom n-ten Grades mit Entwicklungsstelle 0 an!

✎

$f^{(0)}(0) = f(0) = \ln 1 = 0$

$f^{(k)}(0) = (-1)^{k-1} \frac{(k-1)!}{(1+0)^k} = (-1)^{k-1} (k-1)!$ für $k = 1, 2, \ldots$

$T_n(x) = \sum\limits_{k=0}^{n} \frac{f^{(k)}(0)}{k!} x^k = \sum\limits_{k=1}^{n} \frac{(-1)^{k-1} (k-1)!}{k!} x^k = \sum\limits_{k=1}^{n} \frac{(-1)^{k-1}}{k} x^k = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} \pm \ldots$

✐

(iii) Geben Sie das zugehörige Restglied $R_n(x, \xi)$ an und zeigen Sie, dass es für $x \in [0, 1]$ gegen 0 geht!

✎

$R_n(x, \xi) = \frac{f^{(n+1)}(\xi)}{(n+1)!} x^{n+1} = (-1)^n \frac{n!}{(n+1)!} \frac{x^{n+1}}{(1+\xi)^{n+1}} = \frac{(-1)^n}{n+1} \left(\frac{x}{1+\xi}\right)^{n+1} \xrightarrow[n\to\infty]{} 0,$
da $\frac{x}{1+\xi} \in [0, 1]$ für $x \in [0, 1]$ und $\xi \in (0, x)$

✐

(iv) Welche Reihe ergibt sich konkret für $x = 1$?

✎

$\ln 2 = f(1) = \sum\limits_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} 1^k = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} \pm \ldots,$
d. h. der Wert der alternierenden harmonischen Reihe ist $\ln 2$.

✐

c) Implementieren Sie eine Java-Methode zur Berechnung des Taylorpolynoms $\sum\limits_{k=0}^{n} \frac{f_k}{k!} x^k$ mit $f_k = f^{(k)}(0) = k$-te Ableitung einer gegebenen Funktion f an der Stelle 0! Als Parameter erhält die Methode ein Array mit den Werten f_k sowie die Stelle x. Achten Sie darauf, unnötige wiederholte Berechnungen zu vermeiden!

Berechnen Sie damit näherungsweise z. B. $e^x = \sum\limits_{k=0}^{\infty} \frac{x^k}{k!}$ für $x = 1$ (d. h. die Eulersche Zahl e, die in Java als Konstante `Math.E` zur Verfügung steht) sowie $\sin x = \sum\limits_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$ für $x = \pi$ (`Math.PI`) und $x = \frac{\pi}{2}$!

✎

```java
// Taylorpolynome.
class Taylor {
    // Taylorpolynom einer Funktion an der Stelle x berechnen.
    // f[k] enthält den Wert der k-ten Ableitung der Funktion
    // an der Entwicklungsstelle 0.
    // Der Grad des Polynoms ist f.length - 1.
    // (Daher muss f mindestens ein Element enthalten.)
    public static double compute (double [] f, double x) {
        // Grad des Polynoms.
        int n = f.length - 1;

        // Summanden s[k] = f[k] * (x hoch k) / (Fakultät von k)
        // berechnen.
        double [] s = new double [n + 1];
        s[0] = f[0];
        double xk = 1;
        for (int k = 1; k <= n; k++) {
            xk = xk * x / k;
            s[k] = f[k] * xk;
        }

        // Die Summanden s[k] aufsummieren.
        // Wenn man mit den kleinen Werten beginnt,
        // erreicht man tendenziell eine höhere Genauigkeit.
        // (Wenn die Schleife andersherum laufen würde,
        // könnte man sie mit der vorigen Schleife zusammenfassen
        // und bräuchte das Hilfsarray s nicht.)
        double y = 0;
        for (int k = n; k >= 0; k--) y += s[k];
        return y;
    }

    // Taylorpolynom n-ten Grades der Exponentialfunktion
    // an der Stelle x berechnen.
    public static double exp (int n, double x) {
        // Die k-te Ableitung der Exponentialfunktion an der Stelle 0
        // ist 1 für jeden Wert von k.
        double [] f = new double [n+1];
        for (int k = 0; k <= n; k++) f[k] = 1;
        return compute(f, x);
    }
```

```
    // Taylorpolynom n-ten Grades der Sinusfunktion
    // an der Stelle x berechnen.
    public static double sin (int n, double x) {
        // Die k-te Ableitung der Sinusfunktion an der Stelle 0
        // ist 0, 1, 0, -1, ... für k = 0, 1, 2, 3, ...
        double [] f = new double [n+1];
        int sign = 1;
        for (int k = 1; k <= n; k += 2) {
            f[k] = sign;
            sign = -sign;
        }
        return compute(f, x);
    }

    // Test.
    // Der Grad n wird als Kommandozeilenargument übergeben.
    public static void main (String [] args) {
        int n = Integer.parseInt(args[0]);

        // Eulersche Zahl e mit maximal möglicher Genauigkeit.
        System.out.println("e =            " + Math.E);

        // Eulersche Zahl e als Näherungswert der Exponentialfunktion
        // an der Stelle 1.
        System.out.println("exp(n, 1) =    " + exp(n, 1));

        // Näherungswerte für sin(PI/2) = 1 und sin(PI) = 0.
        System.out.println("sin(n, PI/2) = " + sin(n, Math.PI/2));
        System.out.println("sin(n, PI) =   " + sin(n, Math.PI));
    }
}
```