Leonhard Fellermayr
Mat.Nr. 221281305090
Tutorium E (Do 14-16 Linde/Wugalter E06)

OCAML

Übungen zur Analysis I für Informatiker - Blatt 12 (fak. bis 02.02.04)

Aufgabe 51. Ableitungen

(a) $f(x) = a^{\sqrt{x^2+1}} = a^{(x^2+1)^{\frac{1}{2}}} = e^{(x^2+1)^{\frac{1}{2}} \cdot \ln(a)}$

$$f'(x) = a^{\sqrt{x^2+1}} \cdot \frac{1}{2 \cdot \sqrt{x^2+1}} \cdot 2x \cdot \ln(a) = a^{\sqrt{x^2+1}} \cdot \frac{x \cdot \ln(a)}{\sqrt{x^2+1}}$$

(b) $f(x) = \sqrt[n]{\dfrac{x}{\ln x}} = \left(\dfrac{x}{\ln(x)}\right)^{\frac{1}{n}}$

$$f'(x) = \frac{1}{n} \cdot \left(\frac{x}{\ln(x)}\right)^{\frac{1-n}{n}} \cdot \frac{\ln(x) - 1}{\ln^2(x)}$$

(c) $f(x) = x^{(x^x)} = e^{(x^x) \cdot \ln x}$

$$f'(x) = \left(e^{x^x \cdot \ln(x)}\right)' = e^{x^x \cdot \ln(x)} \cdot (x^x \cdot \ln(x))' = x^{(x^x)} \cdot \left[(x^x)' \cdot \ln(x) + x^x \cdot \frac{1}{x}\right] =$$

$$= x^{(x^x)} \cdot \left[\left(e^{x \cdot \ln x}\right)' \cdot \ln(x) + x^x \cdot \frac{1}{x}\right] == x^{(x^x)} \cdot \left[x^x \cdot (\ln(x)+1) \cdot \ln(x) + x^x \cdot \frac{1}{x}\right] =$$

$$= x^{(x^x)} \cdot \left[x^x \cdot \ln^2(x) + x^x \cdot \ln(x) + x^{x-1}\right] = x^{(x^x)} \cdot x^x \cdot \left[\ln^2(x) + \ln(x) + \frac{1}{x}\right]$$

(d) $f(x) = \arccos(x)$. Gemäß **Satz 5.7**[1] gilt für die Ableitung:

$$f'(x) = \frac{1}{\cos'(\arccos x)} = \frac{1}{-\sin(\arccos x)} =^{(*)} -\frac{1}{\sqrt{1-x^2}}$$

(*): Für $y := \arccos x$ gilt $\cos y = x$. Wegen

$$\sin^2 y + \cos^2 y = 1 \Leftrightarrow \sin y = \sqrt{1 - \cos^2 y} = \sqrt{1-x^2}$$

ist

$$-\sin y = -\sin(\arccos x) = -\sqrt{1-x^2}.$$

Aufgabe 52. $f(x) = x^{n+1} - rx^n + r - 1$

(a) **Ableitung:**

$$f'(x) = (n+1)x^n - nrx^{n-1} = x(n+1)x^{n-1} - nrx^{n-1}$$

[1] vgl. auch Forster, Analysis I, S.149

Extrema (Nullstellen der Ableitung):

$$\begin{aligned} x(n+1)x^{n-1} - nrx^{n-1} &= 0 \\ x(n+1)x^{n-1} &= n \cdot r \cdot x^{n-1} \\ x(n+1) &= n \cdot r \\ x &= \tfrac{n \cdot r}{n+1} \text{ (Extrempunkt)} \end{aligned}$$

$$r > 1 + \frac{1}{n} \Rightarrow \frac{n \cdot r}{n+1} > \frac{n \cdot \left(1 + \frac{1}{n}\right)}{n+1} = 1$$

Außerdem:

$x > \frac{n \cdot r}{n+1} \Rightarrow f'(x) > 0$

$x < \frac{n \cdot r}{n+1} \Rightarrow f'(x) < 0$

$\Rightarrow$ Bei dem Extrempunkt $\frac{n \cdot r}{n+1}$ muss es sich um ein Minimum handeln.

(b) **Bestimmung der Nullstellen:**

$$\begin{aligned} & x^{n+1} - rx^n + r - 1 = 0 \\ \Leftrightarrow \quad & x^{n+1} - 1 = rx^n - r \qquad \Rightarrow \text{1. (triviale) Lösung: } x = 1 \\ \Leftrightarrow \quad & x^{n+1} - 1 = r(x^n - 1) \\ \Leftrightarrow \quad & \tfrac{x^{n+1}-1}{x^n-1} = r \\ \Leftrightarrow \quad & \tfrac{x^{n+1}-1}{x^n-1} > 1 + \tfrac{1}{n} \\ \Leftrightarrow \quad & \tfrac{x \cdot (x^n-1)}{x^n-1} > 1 + \tfrac{1}{n} \\ \Leftrightarrow \quad & x > 1 + \tfrac{1}{n} \qquad \Rightarrow \text{2. Lösung: } x > \tfrac{1}{n} \end{aligned}$$

Aufgabe 53.

(a) **Grenzwerte:**

$$\lim f(x) = \lim x^{\frac{1}{x}} = \lim e^{\frac{1}{x} \cdot ln(x)} = \lim e^{\frac{ln(x)}{x}}$$

$$\lim_{x \to 0} e^{\frac{\ln(x)}{x}} = e^{-\infty} = \frac{1}{e^{+\infty}} = 0$$

$$\lim_{x \to \infty} e^{\frac{\ln(x)}{x}} = e^0 = 1$$

Ableitung von $\sqrt[x]{x} = x^{\frac{1}{x}} = e^{\frac{1}{x} \cdot \ln(x)}$ über *Kettenregel*:

$$f'(x) = exp'\left(\frac{1}{x} \cdot \ln(x)\right) \cdot \left[\left(-\frac{1}{x^2} \cdot \ln(x)\right) + \left(\frac{1}{x} \cdot \frac{1}{x}\right)\right] = \exp\left(\frac{1}{x} \cdot \ln(x)\right) \cdot \left[\frac{-\ln(x)}{x^2} + \frac{1}{x^2}\right] =$$

$$\exp\left(\frac{1}{x} \cdot \ln(x)\right) \cdot \left[\frac{1 - \ln(x)}{x^2}\right] = x^{\frac{1}{x}} \cdot \frac{1 - \ln(x)}{x^2}$$

Extremum: $f'(x) = 0 \Leftrightarrow x = e$

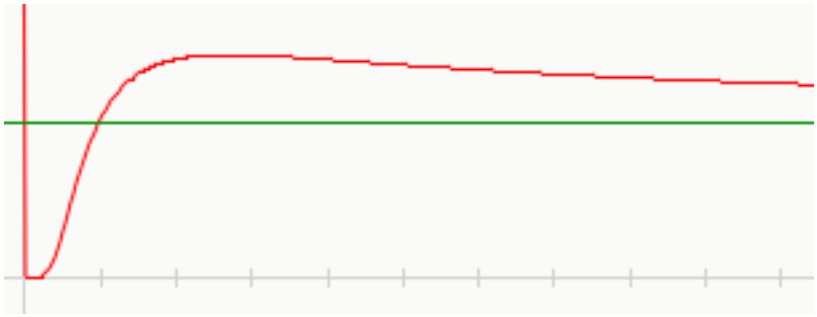
Maximum:

$\lim_{x \nearrow e} \left[\sqrt[x]{x} \cdot \frac{1-\ln(x)}{x^2}\right] \geq 0$ $\qquad$ $\lim_{x \searrow e} \left[\sqrt[x]{x} \cdot \frac{1-\ln(x)}{x^2}\right] \leq 0$

Damit hat f tatsächlich[2] ein Maximum an der Stelle e.

[2] vgl. Forster, Analysis I, S. 154

(b)(c) Stelle den Bezug zur Funktion $f = x^{\frac{1}{x}} = \sqrt[x]{x}$ her:



$$m^n = n^m \Rightarrow m^{n\frac{1}{m}} = n^{m\frac{1}{m}} \Rightarrow \left(m^{\frac{n}{m}}\right)^{\frac{1}{n}} \Rightarrow m^{\frac{1}{m}} = n^{\frac{1}{n}}$$

Die Lösung der Gleichung mit $m = n$ ist trivial.

Eine Lösung (m,n) mit $m \neq n$ muss auf jeden Fall im Intervall $I =]1; +\infty[$ liegen, da *nur hier* die Zuordnung $y := x$ mehrdeutig ist, es also nur hier für ein und dasselbe y **zwei** Lösungen gibt. Wegen des Zwischenwertsatzes muss außerdem entweder $(m > e, n < e)$ oder $(m < e, n > e)$ gelten (da e Maximum der Funktion). Damit kommt als ganzzahlige Lösung (m,n) nur $(2, 4)$ bzw. $(4, 2)$ in Frage, wie aus dem Graphen von f ersichtlich wird (denn: $1 \notin I$, $3 > e$).

Aufgabe 54.

Für $n \in \mathbb{N}_0$, $x \in [-1, 1]$ sei $T_n(x) = \cos(n \cdot \arccos(x))$ und $U_n(x) = \frac{\sin((n+1)\arccos(x))}{\sqrt{1-x^2}}$.

(a) Zu zeigen: $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$, umgeformt $T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x)$.

Im Folgenden sei abkürzend $y := \arccos(x)$.

$$T_{n+1}(x) + T_{n-1}(x) = \cos((n+1) \cdot \arccos(x)) + \cos((n-1) \cdot \arccos(x)) =$$

$$= \cos(ny + y) + \cos(ny - y) =$$

$$= \cos(ny)\cos(y) - \sin(ny)\sin(y) + \cos(ny)\cos(y) + \sin(ny)\sin(y) =$$

$$= 2 \cdot \cos(ny)\cos(y) = 2 \cdot \cos(n \cdot \arccos(x)) \cdot \cos(\arccos(x)) = 2 \cdot x \cdot \cos(n \cdot \arccos(x)) =$$

$$= 2 \cdot x \cdot T_n(x)$$

Berechne T_0, T_1:

$$T_0 = \cos(0 \cdot \arccos(x)) = \cos(0) = 1$$

$$T_1 = \cos(1 \cdot \arccos(x)) = \cos(\arccos(x)) = x$$

Wegen der oben bewiesenen Beziehung $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ gilt $grad(T_n) < grad(T_{n+1})$, genauer $grad(T_n) + 1 = grad(T_{n+1})$. Mit $grad(T_0) = 0$ ist stets $grad(T_n) = n$.

(b) Nullstellen:

$$\begin{aligned} & cos(n \cdot \arccos(x)) &=& \ 0 \\ \Leftrightarrow \quad & n \cdot \arccos(x) &=& \ \pm\tfrac{\pi}{2} \\ \Leftrightarrow \quad & \arccos(x) &=& \ \pm\tfrac{\pi}{2n} \\ \Leftrightarrow \quad & x_{1,2} &=& \ \cos\left(\pm\tfrac{\pi}{2n}\right) \end{aligned}$$

(c) *Anmerkung:* $\arccos'(x)$ wurde in Aufgabe 51d) bestimmt.

$$T_n(x) = \cos(n \cdot \arccos(x))$$

$$T_n'(x) = -\sin(n \cdot \arccos(x)) \cdot n \cdot \left(-\frac{1}{\sqrt{1-x^2}}\right) = \frac{\sin(n \cdot \arccos(x))}{\sqrt{1-x^2}} \cdot n = U_n \cdot n$$

Für $n := n+1$ ergibt sich tatsächlich $T'_{n+1} = U_n \cdot (n+1)$.

Der Grad der 1. Ableitung f' ist stets um 1 geringer als der Grad der Ursprungsfunktion f, also $grad(f') = grad(f) - 1$. Damit (und nach der Feststellung in 54a) ist $grad\left[T'_{n+1}\right] = n$ und wegen der oben bewiesenen Beziehung auch $grad(U_n) = n$.

Aufgabe 55. Newton-Maehly-Approximation, Horner-Schema in OCAML

Programm-Ausgabe:

```
(a) | m | Nullstelle  | # Iterationen | Genauigkeit eingehalten
    ---------------------------------------------------------
    | 0 |  3.00000000 | 010           | false
    | 1 |  2.00000000 | 008           | false
    | 2 |  1.00000000 | 001           | false

(b) | m | Nullstelle  | # Iterationen | Genauigkeit eingehalten
    ---------------------------------------------------------
    | 0 |  0.53208889 | 007           | false
    | 1 | -0.65270364 | 100           | true
    | 2 | -2.87938524 | 100           | true

(c) | m | Nullstelle  | # Iterationen | Genauigkeit eingehalten
    ---------------------------------------------------------
    | 0 |  4.00355758 | 014           | false
    | 1 |  1.98318161 | 013           | false
    | 2 |  0.04073553 | 012           | false
    | 3 | -1.03226355 | 100           | true
    | 4 | -2.99521117 | 100           | true
```

Programm-Quelltext[3]:

```
(* common exception                                                    *)

exception MyError of string ;;

(* max_of_list : get max value from list                               *)

let rec max_of_list = function
    []       -> raise (MyError "Leere Liste hat kein maximales Element!")
  | x::[]    -> x
  | x::y::xs -> max_of_list ((max x y)::xs) ;;

(* strip_first : strip first element from list                         *)

let strip_first = function
    []    -> []
  | x::xs -> xs ;;

(* strip_last : strip last element from list (tail recursive version)  *)

let strip_last =
  let rec sl a = function
```

[3] Auch im WWW: http://www.slacky.de/files/uni/analysis/newton.ml

```
    []    -> []
  | x::[] -> a
  | x::xs -> sl (a@[x]) xs
  in sl [] ;;

(* lastele : return last element of list                                    *)

let rec lastele = function
    []    -> raise (MyError "Leere Liste hat kein letztes Element!")
  | x::[] -> x
  | x::xs -> lastele xs ;;

(* get_ele_nr : gets element nr. x from list l, counting from 0            *)

let rec get_ele_nr s = function
    []    -> raise (MyError "Leere Liste enthaelt keine Elemente!")
  | x::xs -> if s = 0 then x else get_ele_nr (s-1) xs ;;

(* implementation of the horner scheme                                     *)

let it_horner x l it =
  let gen_hornerline x l =
    let l = strip_first l in
    let rec fh_aux lf acc x = function
      []   -> acc@[lf]
    | h::t -> fh_aux (h +. x *. lf) (acc@[lf]) x t
    in fh_aux 1. [] x l
  in
  let rec iterate x l itcur =
    let l = if itcur > 1 then strip_last l else l in
      if itcur < it then iterate x (gen_hornerline x l) (itcur+1)
      else lastele l
  in iterate x l 0 ;;

(* Bestimme x0 := max (|a0|,1+|a1|,...,1+|a{n-1}|) aus Koeffizientenliste *)

let bestimme_x0 l =
  let a0 = lastele l and rest = strip_first (strip_last l) in
  max_of_list ([abs_float a0]@(List.map (fun y -> 1.+.(abs_float y)) rest)) ;;

(* newton_maehly : Implementation                                          *)

let newton_maehly klist grad epsilon itmax =
  let x0 = (bestimme_x0 klist) in
  let rec calc m nslist =
    let rec nst_approx xk itcur =
       let pxk = it_horner xk klist 1 and p_xk = it_horner xk klist 2
         and extract_nst (nst,it,gen,m) = nst in
       let rec theSum k n =
          if k > n then 0.
          else let theNst = extract_nst (get_ele_nr (k-1) nslist) in
               pxk /. (xk -. theNst) +. theSum (k+1) n in
       let xkplus1 = xk -. pxk /. ( p_xk -. (theSum 1 m) ) in
       let genau = abs_float (xkplus1 -. xk) > epsilon *. xkplus1 in
       if itcur = itmax || not genau then (xkplus1,itcur,genau,m)
       else nst_approx xkplus1 (itcur+1) in
    if (m < grad) then calc (m+1) (nslist@[nst_approx x0 0])
    else nslist
  in calc 0 [] ;;

(* Ausgabefunktion                                                         *)

let pp = Printf.printf;;
```

```
let output_example x =
  pp "| m | Nullstelle  | # Iterationen | Genauigkeit eingehalten\n" ;
  pp "-------------------------------------------------------------\n" ;
  List.iter (fun (a,b,c,d) -> pp "| %d | % .8f | %03d           | %b\n" d a b c) x ;
  pp "\n" ;;

(* Aufruf fuer die Beispiele (a) - (c)                                    *)

output_example (newton_maehly [1.;-6.;11.;-6.] 3 10e-8 100) ;;
output_example (newton_maehly [1.;3.;0.;-1.] 3 10e-8 100) ;;
output_example (newton_maehly [1.;-2.;-13.;14.;24.;-1.] 5 10e-8 100) ;;
```