Matthias Jauernig
03IN1

17.10.04

Numerische Mathematik

1. Beleg

Rahmenbedingungen:

Alle hier verwendeten Programme und Ergebnisausdrucke stützen sich auf eine Kompilierung mittels des C-Compilers der GNU Compiler Collection unter Linux auf einem AMD Athlon XP Prozessor (32 Bit). Die verwendeten Funktionen entstammen dem ANSI-C-Standard und der Implementation der glibc. Es wurde darauf geachtet, dass nicht nur die Endergebnisse, sondern auch Zwischenwerte auf jeden Fall im float-Format (single precision) vorliegen, wenn nötig musste dies durch explizites Casten oder zusätzliche Variablen erreicht werden.

1.) Näherungen für die erste Ableitung einer Funktion:

- Ergebnisausdruck: *(Quelltext im Anhang)*

```
1.) f(x)=e^x
------------
f'(1)=2.718282
h               d1f             |d1f-f'(1)|     d2f             |d2f-f'(1)|
1.00e+00        4.6707745       1.9524927       3.1945281       0.4762464
3.16e-01        3.1972058       0.4789240       2.7638137       0.0455320
1.00e-01        2.8588438       0.1405621       2.7228153       0.0045335
3.16e-02        2.7617185       0.0434368       2.7187324       0.0004506
1.00e-02        2.7319193       0.0136375       2.7183175       0.0000358
3.16e-03        2.7225735       0.0042918       2.7182760       0.0000057
1.00e-03        2.7198789       0.0015972       2.7183292       0.0000474
3.16e-04        2.7194824       0.0012007       2.7183516       0.0000699
1.00e-04        2.7203560       0.0020742       2.7191639       0.0008821
3.16e-05        2.7217441       0.0034623       2.7179744       0.0003073
1.00e-05        2.7418137       0.0235319       2.7298927       0.0116110
3.16e-06        2.7895994       0.0713177       2.7519021       0.0336204
1.00e-06        2.6226044       0.0956774       2.6226044       0.0956774
3.16e-07        3.0157831       0.2975013       2.6388102       0.0794716
1.00e-07        4.7683716       2.0500898       3.5762787       0.8579969
3.16e-08        0.0000000       2.7182817       0.0000000       2.7182817
1.00e-08        0.0000000       2.7182817       0.0000000       2.7182817
3.16e-09        0.0000000       2.7182817       0.0000000       2.7182817
1.00e-09        0.0000000       2.7182817       0.0000000       2.7182817
3.16e-10        0.0000000       2.7182817       0.0000000       2.7182817
1.00e-10        0.0000000       2.7182817       0.0000000       2.7182817
3.16e-11        0.0000000       2.7182817       0.0000000       2.7182817
1.00e-11        0.0000000       2.7182817       0.0000000       2.7182817


2.) f(x)=1/((x+0.13)(x-0.13))
-----------------------------
f'(1)=-2.069353
h               d1f             |d1f-f'(1)|     d2f             |d2f-f'(1)|
1.00e+00        -0.7661299      1.3032233       29.7113304      31.7806835
3.16e-01        -1.3733425      0.6960106       -2.5869687      0.5176156
1.00e-01        -1.7903787      0.2789744       -2.1136119      0.0442588
3.16e-02        -1.9731308      0.0962223       -2.0737157      0.0043626
1.00e-02        -2.0380378      0.0313153       -2.0697892      0.0004361
3.16e-03        -2.0593653      0.0099878       -2.0693927      0.0000396
1.00e-03        -2.0662544      0.0030987       -2.0693538      0.0000007
3.16e-04        -2.0684502      0.0009029       -2.0693927      0.0000396
1.00e-04        -2.0694733      0.0001202       -2.0694733      0.0001202
3.16e-05        -2.0695810      0.0002279       -2.0676961      0.0016570
1.00e-05        -2.0742416      0.0048885       -2.0682812      0.0010719
3.16e-06        -2.0733509      0.0039978       -2.0545022      0.0148509
1.00e-06        -2.1457672      0.0764141       -2.0861626      0.0168095
3.16e-07        -2.2618372      0.1924841       -2.0733509      0.0039978
1.00e-07        -2.3841858      0.3148327       -1.7881393      0.2812138
3.16e-08        -3.7697289      1.7003758       -1.8848644      0.1844887
1.00e-08        -11.9209290     9.8515759       -5.9604645      3.8911114
3.16e-09        -37.6972885     35.6279354      -18.8486443     16.7792912
1.00e-09        0.0000000       2.0693531       0.0000000       2.0693531
3.16e-10        0.0000000       2.0693531       0.0000000       2.0693531
1.00e-10        0.0000000       2.0693531       0.0000000       2.0693531
3.16e-11        0.0000000       2.0693531       0.0000000       2.0693531
1.00e-11        0.0000000       2.0693531       0.0000000       2.0693531
```

- Analyse:

  Für $f(x)=e^x$ liefert der *vordere Differenzenquotient* mit 2 richtigen Nachkommastellen bei $h=10^{-7/2}$ das genaueste Ergebnis als Näherung der Ableitung f'(1), für den *zentralen Differenzenquotienten* ist dies bei $h=10^{-5/2}$ mit 3 richtigen Nachkommastellen der Fall.

  Für $f(x)=\frac{1}{(x+0.13)\,(x-0.13)}$ ist die Näherung für f'(1) mit 3 korrekten Nachkommastellen durch den *vorderen Differenzenquotienten* bei $h=10^{-9/2}$ am genauesten, beim *zentralen Differenzenquotienten* werden 6 richtige Nachkommastellen bei $h=10^{-3}$ ausgegeben.

  Theoretisch müsste man erwarten, dass sich mit kleinerer werdender Abweichung h die Näherungswerte für die Ableitung dem tatsächlichen Ergebnis immer weiter annähern würden, aus der Sekante immer weiter eine Tangente durch den Punkt (x,f(x)) wird. Dem ist aber nicht so, was auf die Verwendung von Rechenanlagen und der damit verbundenen Maschinengenauigkeit zurück zu führen ist.

  Für $f(x)=e^x$ liefern beide Näherungsverfahren ab $h=10^{-15/2}$ den Wert 0 für f'(1), was offensichtlich falsch ist. Grund dafür ist die Berechnung von e^{1+h}, e^1 und e^{1-h}, welche für sehr kleine h mit „single precision" denselben Wert ergeben, da 1+h und 1-h für ein h mit mehr als 7 Nullen im Nachkommaanteil (0.0000000....) gleich 1 ist (Rundung auf float). Für $h>10^{-15/2}$ sieht die Berechnung durch die beiden Näherungsverfahren so aus, dass die Abweichung vom tatsächlichen Ableitungswert zunächst ein Minimum erreicht, um dann wieder anzuwachsen, wobei der zentrale Differenzenquotient offensichtlich einen genaueren Wert abliefert.

  Genau dasselbe Verhalten beobachtet man bei $f(x)=\frac{1}{(x+0.13)\,(x-0.13)}$. Zunächst erreichen die Abweichungen beider Differenzenquotienten ebenfalls ein Minimum, bevor ihre Werte wieder stark abweichen aufgrund der verwendeten Maschinengenauigkeit bei float. Ab h=10-9 liefern die beiden Näherungsverfahren wieder den Wert 0 für f'(1) und wieder aufgrund der Verwendung von float-Zahlen. Dadurch stimmen x+h und x-h mit dem float-Wert von x überein, sodass f(x+h) dasselbe ergibt wie f(x-h) und f(x), wodurch der hintere Faktor jeweils 0 wird und damit der ganze Term. Auch bei dieser Funktion gilt: der zentrale Differenzenquotient ist aufgrund seiner Berechnung genauer als der vordere Differenzenquotient und ist diesem somit vorzuziehen.

2.) <u>Integrale:</u>

(a) Gültigkeit der Ungleichungen zeigen:

1.) $\rightarrow 0<I(k+1) \wedge 0<I(k)$ **erfüllt, da sowohl** $\frac{1}{e}>0$, $e^x>0$ **sowie** $x^k>0\ (x^{k+1}>0)$ **für** $x\in[0,1]$

$\rightarrow I(k+1)=\frac{1}{e}\int_0^1 e^x x^{k+1}\,dx < I(k)=\frac{1}{e}\int_0^1 e^x x^k\,dx,$

da $x^{k+1}<x^k$ **für** $x\in(0,1)$ **und** $x^{k+1}=x^k$ **für** $x\in\{0,1\}$

$\rightarrow$ **q.e.d.**

2.) $\frac{1}{e*(k+1)} < I(k) < \frac{1}{k+1}$

$\rightarrow I(k)=\frac{1}{e}\int_0^1 e^x x^k\,dx < \frac{1}{e}\int_0^1 e^1 x^k\,dx=\int_0^1 x^k\,dx=\left[\frac{x^{k+1}}{k+1}\right]_0^1=\frac{1}{k+1} \rightarrow I(k)<\frac{1}{k+1}$

$\rightarrow I(k)=\frac{1}{e}\int_0^1 e^x x^k\,dx > \frac{1}{e}\int_0^1 e^0 x^k\,dx=\frac{1}{e}\int_0^1 x^k\,dx=\frac{1}{e}\left[\frac{x^{k+1}}{k+1}\right]_0^1=\frac{1}{e*(k+1)} \rightarrow I(k)>\frac{1}{e*(k+1)}$

(b) Partielle Integration:

$$I(k+1) = \frac{1}{e} * \left(\left[e^x x^{k+1} \right]_0^1 - (k+1) * \int_0^1 e^x x^k \, dx \right) = 1 - (k+1) * I(k)$$

$$\rightarrow I(k+1) + (k+1) * I(k) ?1 - (k+1) * I(k) + (k+1) * I(k) = 1 \rightarrow q.e.d.$$

(c) Rekursionsmethoden zur näherungsweisen Integralbestimmung:

- Ergebnisausdruck: *(Quelltext im Anhang)*

k	(A): I(k)	(B): I(k)
0	0.6321205	0.6321205
1	0.3678795	0.3678795
2	0.2642411	0.2642411
3	0.2072767	0.2072766
4	0.1708932	0.1708934
5	0.145534	0.1455330
6	0.1267958	0.1268024
7	0.1124296	0.1123835
8	0.100563	0.1009320
9	0.09493256	0.0916123
10	0.05067444	0.0838771
11	0.4425812	0.0773522
12	-4.310974	0.0717733
13	57.04266	0.0669477
14	-797.5973	0.0627322
15	11964.96	0.0590175
16	-191438.3	0.0557193
17	3254453	0.0527711
18	-5.858015e+07	0.0501199
19	1.113023e+09	0.0477228
20	-2.226046e+10	0.0455449
21	4.674696e+11	0.0435574
22	-1.028433e+13	0.0417364
23	2.365396e+14	0.0400618
24	-5.67695e+15	0.0385166
25	1.419238e+17	0.0370862
26	-3.690018e+18	0.0357584
27	9.963048e+19	0.0345225
28	-2.789653e+21	0.0333693
29	8.089995e+22	0.0322907
30	-2.426999e+24	0.0312797

Mit gestörten Startwerten:

k	(A): I(k)	(B): I(k)
0	0.6321216	0.6321205
1	0.3678784	0.3678795
2	0.2642431	0.2642411
3	0.2072706	0.2072766
4	0.1709175	0.1708934
5	0.1454124	0.1455330
6	0.1275253	0.1268024
7	0.1073227	0.1123835
8	0.1414185	0.1009320
9	-0.2727661	0.0916123
10	3.727661	0.0838771
11	-40.00427	0.0773522
12	481.0513	0.0717733
13	-6252.667	0.0669477
14	87538.33	0.0627322
15	-1313074	0.0590175
16	2.100918e+07	0.0557193
17	-3.571561e+08	0.0527711
18	6.42881e+09	0.0501199
19	-1.221474e+11	0.0477228
20	2.442948e+12	0.0455449
21	-5.130191e+13	0.0435574
22	1.128642e+15	0.0417364
23	-2.595877e+16	0.0400618
24	6.230104e+17	0.0385166
25	-1.557526e+19	0.0370862
26	4.049567e+20	0.0357584
27	-1.093383e+22	0.0345225
28	3.061473e+23	0.0333693
29	-8.878272e+24	0.0322907
30	2.663482e+26	0.0312797

- Analyse:

  Offensichtlich gilt für Eingangs- und Rundungsfehler bei Algorithmus (A) eine Fortpflanzung mit dem Faktor *k*, wodurch schon bei kleinen k sehr große Fehler entstehen.
  Anders sieht es bei (B) aus: hier pflanzen sich die Fehler mit dem Faktor *1/k* fort, sodass bei größerem k sogar noch korrigiert wird und dieser Algorithmus damit auf jeden Fall dem ersten vorzuziehen ist.

  Stört man den Startwert bei Algorithmus (A) um $+10^{-6}$, so hat dies beträchtlichen Einfluss auf das Ergebnis I(k). Ist bei I(0) erst die 7. Nachkommastelle gestört, handelt es sich bei I(4) hier schon um die 4. Nachkommastelle, bei I(8) um die 2. und ab I(9) scheinen die Ergebnisse komplett abzuweichen. Dies kann man wieder auf die Fehlerfortpflanzung mit dem Faktor k zurückführen – schon kleine Abweichungen vom Startwert führen hier zu großen Änderungen am Ergebnis, vor allem für große k.
  Stört man hingegen bei (B) den Startwert um den vergleichsweise großen Wert $+10^{-1}$, so hat dies keine Abweichungen in den dargestellten Ziffern zur Folge – sie stimmen mit denen des „ungestörten" Algorithmus' überein. Grund dafür ist der Faktor *1/k*, durch den Fehler noch „korrigiert" werden können und durch den die relativ große Abweichung keine Auswirkungen auf das Endergebnis hat.

3.) Quadratische Gleichung:

- Ergebnisaudruck: *(Quelltext im Anhang)*

p	x1,x1'	x2	x2'
1e+00	-2.06	0.05992985	0.05992983
1e+01	-20	0.006170273	0.006170658
1e+02	-200	0.000617981	0.0006172543
1e+03	-2e+03	6.103516e-05	6.172562e-05
1e+04	-2e+04	0	6.172562e-06
1e+05	-2e+05	0	6.172561e-07
1e+06	-2e+06	0	6.172561e-08
1e+07	-2e+07	0	6.172562e-09
1e+08	-2e+08	0	6.172562e-10
1e+09	-2e+09	0	6.172562e-11
1e+10	-2e+10	0	6.172562e-12
1e+11	-2e+11	0	6.172562e-13
1e+12	-2e+12	0	6.172562e-14
1e+13	-2e+13	0	6.172562e-15
1e+14	-2e+14	0	6.172561e-16
1e+15	-2e+15	0	6.172562e-17

- Analyse:

  x_2 liefert ab p=10000 schon kein brauchbares Ergebnis mehr (x_2=0), wohingegen $\overline{x_2}$ bis zum Endwert $p=10^{15}$ einen korrekten Wert liefert.

  Dies ist damit zu begründen, dass bei der Berechnung von x_2 mittels quadratischer Lösungsformel das kleine q ab einem hinreichend großen p nicht mehr in das Ergebnis von p^2+q einfließt (Auslöschungsfehler tritt auf), da mit „single precision" (float) gearbeitet wird, bei dem nur maximal 8 relevante Dezimalstellen gespeichert werden können. Da bei p=10000 schon p^2=100000000 ist, also 9 relevante Stellen aufweist, kann der kleine Wert von q nicht mehr mit in die Rechnung einfließen, sodass die Wurzel aus der Summe p^2+q bei Verwendung von float gleich p ist und das Endergebnis für x_2 logischerweise 0 (p-p=0).

  Anders sieht es bei der Berechnung von $\overline{x_2}$ mit Hilfe des *Vieta'schen Wurzelsatzes* ($\overline{x_2}$=-q/$\overline{x_1}$) aus. Da $\overline{x_1}$ nahezu korrekt berechnet wurde (Ergebnis ist 2p, da auch hier q nicht mehr mit einfließt), kann man auf dessen Ergebnis zurückgreifen und mit Hilfe der Beziehung von Vieta $\overline{x_2}$ berechnen. Hier kommt es bei der Division zu keiner Vernachlässigung von q, woraufhin dieser Term wegfallen würde, wodurch ein korrektes Ergebnis geliefert werden kann.

Anhang: Quelltexte

Zu 1.)

```
#include <stdio.h>
#include <math.h>

int main(void){
  float h, j, fabl, d1f, d2f, zw1, zw2, zw3;

  printf("\n1.) f(x)=e^x\n------------\nf'(1)=%f\n",fabl=expf(1));
  printf("h\t\td1f\t\t|d1f-f'(1)|\td2f\t\t|d2f-f'(1)|\n");
  for(j=0.0,h=1.0; j>=-11; h=powf(10,j-=0.5)){
        zw1=1/h;      zw2=expf(1+h);              zw3=expf(1);
        d1f=zw1*(zw2-zw3);
        zw1=1/(2*h);  zw2=expf(1+h);              zw3=expf(1-h);
        d2f=zw1*(zw2-zw3);
        printf("%3.2e\t%8.7f\t%8.7f\t%8.7f\t%8.7f\n",
               h,d1f,fabsf(d1f-fabl),d2f,fabsf(d2f-fabl));
  }

  printf("\n\n2.) f(x)=1/((x+0.13)(x-0.13))\n"
         "----------------------------\nf'(1)=%f\n",
         fabl=-2/(0.9831f*0.9831f));
  printf("h\t\td1f\t\t|d1f-f'(1)|\td2f\t\t|d2f-f'(1)|\n");
  for(j=0.0,h=1.0; j>=-11; h=powf(10,j-=0.5)){
        zw1=1/h;      zw2=1/((1+h+0.13f)*(1+h-0.13f));
        zw3=1/((1+0.13f)*(1-0.13f));
        d1f=zw1*(zw2-zw3);
        zw1=1/(2*h);  zw2=1/((1+h+0.13f)*(1+h-0.13f));
        zw3=1/((1-h+0.13f)*(1-h-0.13f));
        d2f=zw1*(zw2-zw3);
        printf("%3.2e\t%8.7f\t%8.7f\t%8.7f\t%8.7f\n",h,d1f,
               fabsf(d1f-fabl),d2f,fabsf(d2f-fabl));
  }
  printf("\n\n");
  return 0;
}
```

Zu 2.)

```
#include <stdio.h>
#include <math.h>

float Ia(int k){ return ((k>0) ? (1-k*Ia(k-1)) : ((expf(1)-1)/expf(1))); }
float Ib(int k){ return ((k<40) ? ((1-Ib(k+1))/(k+1)) : (0.7f/41)); }

int main(void){
  int k;

  printf("\n\n k\t\t(A): I(k)\t\t(B): I(k)\n");
  for(k=0; k<=30; k++)
  printf("%2d\t\t%8.7g\t\t%8.7f\n",k,Ia(k),Ib(k));
  printf("\n\n");
  return 0;
}
```

Zu 3.)

```
#include <stdio.h>
#include <math.h>

int main(void){
  int i;
  const float q=0.123451234;
  float p, x1, x21, x22;

  printf("p\t\tx1/x1'\t\tx2\t\tx2'\n");
  for(i=0, p=1.0; i<=15; i++, p*=10.0){
        x1=-p-sqrtf(p*p+q);
        x21=-p+sqrtf(p*p+q);
        x22=-q/x1;
        printf("%1.0e\t\t%6.3g\t\t%8.7g\t%8.7g\n",p,x1,x21,x22);
  }
  printf("\n\n");
  return 0;
}
```