Richardson-Extrapolation

April 24, 2019

Wir wollen den Wert der (ausreichend häufig differenzierbaren) Funktion f an der Stelle $x = 0$ ausrechnen. Dies ist jedoch nicht direkt machbar (etwa, weil f an der Stelle 0 einen Grenzwert repräsentiert). Wir werten deshalb f an einigen Stellen h_k aus (in der Nähe der 0), legen ein Interpolationspolynom hindurch und werten es an der Stelle 0 aus (mit neville).

Wir wählen im Folgenden jeweils $h_k = 2^{-k}$, $k = 0 \dots n$, für verschiedene Funktionen.

```
In [1]: %%capture
        %run Polynominterpolation.ipynb
```

Wir beginnen mit der Berechnung der Ableitung der Exponentialfunktion an der Stelle x. Wir wissen

$$g'(x) = \lim_{h \mapsto 0, h \neq 0} \frac{g(x+h) - g(x)}{h}.$$

Wir rechnen den Differenzenquotionten

$$f(h) = \frac{g(x+h) - g(x)}{h}$$

für die h_k aus. Bemerkung: Da $g \in C^\infty$, ist auch $f \in C^\infty$.

```
In [2]: n=5
        t=[2**(-k) for k in range(0,n+1)]
        def g(x):
            return math.exp(x)
        y=[(g(h)-g(0))/h for h in t]
        print(np.array(y)-math.exp(0))
```

```
[0.71828183 0.29744254 0.13610167 0.06518762 0.03191134 0.01578904]
```

Der Differenzenquotient liefert Näherungen bis 0.015. Wir berechnen das Neville-Schema.

```
In [3]: A=neville_auswertung(t,y,0)
        np.set_printoptions(precision=4)
        print(A-1)
```

```
[[ 7.1828e-01 -1.2340e-01  7.4800e-03 -1.7960e-04  1.8020e-06 -7.7918e-09]
 [-1.0000e+00  2.9744e-01 -2.5239e-02  7.7785e-04 -9.5357e-06  4.8765e-08]
 [-1.0000e+00 -1.0000e+00  1.3610e-01 -5.7264e-03  8.8887e-05 -5.5026e-07]
```

```
 [-1.0000e+00 -1.0000e+00 -1.0000e+00  6.5188e-02 -1.3649e-03  1.0629e-05]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00  3.1911e-02 -3.3326e-04]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00  1.5789e-02]]
```

Aus diesen (schlechten) Approximationen bekommen wir Näherungen bis zur Größenordnung 1e-8.

Wir schauen auf ein zweites Beispiel: Berechnung der Funktion $sinc(x) = \frac{\sin x}{x}$ an der Stelle $x = 0$. Die exakte stetige Fortsetzung hat den Wert 1. Wir versuchen wieder, diesen Wert zu berechnen.

```
In [4]: n=5
        t=[2**(-k) for k in range(0,n+1)]
        y=[math.sin(h)/h for h in t]
        print(np.array(y)-1)
        A=neville_auswertung(t,y,0)
        np.set_printoptions(precision=4)
        print(A-1)
```

```
[-0.1585 -0.0411 -0.0104 -0.0026 -0.0007 -0.0002]
[[-1.5853e-01  7.6231e-02  1.7637e-03 -1.2282e-04 -3.6773e-07  5.8384e-09]
 [-1.0000e+00 -4.1149e-02  2.0381e-02  1.1300e-04 -8.0212e-06 -5.8356e-09]
 [-1.0000e+00 -1.0000e+00 -1.0384e-02  5.1799e-03  7.1061e-06 -5.0680e-07]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -2.6021e-03  1.3003e-03  4.4482e-07]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -6.5091e-04  3.2541e-04]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -1.6275e-04]]
```

Wir starten hier mit deutlich besseren Werten (Fehler 1e-4), aber die Genauigkeit am Ende ist ungefähr dieselbe. Grund dafür: Hier hat der Fehler eine Taylorentwicklung in h^2, und wir müssen dies in t ansetzen.

```
In [5]: n=5
        t=[2**(-k) for k in range(0,n+1)]
        y=[math.sin(h)/h for h in t]
        t2=[h**2 for h in t]
        print(np.array(y)-1)
        A=neville_auswertung(t2,y,0)
        np.set_printoptions(precision=4)
        print(A-1)
```

```
[-0.1585 -0.0411 -0.0104 -0.0026 -0.0007 -0.0002]
[[-1.5853e-01 -2.0222e-03 -3.0442e-06 -6.6472e-10 -2.3759e-14 -1.1102e-16]
 [-1.0000e+00 -4.1149e-02 -1.2924e-04 -4.8220e-08 -2.6202e-12 -1.1102e-16]
 [-1.0000e+00 -1.0000e+00 -1.0384e-02 -8.1229e-06 -7.5602e-10 -1.0325e-14]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -2.6021e-03 -5.0839e-07 -1.1823e-11]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -6.5091e-04 -3.1785e-08]
 [-1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -1.0000e+00 -1.6275e-04]]
```

Jetzt ist die Verbesserung dramatisch.

```
In [ ]:
```