Praktikum 1

Simon Stingelin

15.04.2023

Inhaltsverzeichnis

1 Einführung in Python3

1.1 Lernziele

1. Sie haben eine funktionierende python3 Installation und können jupyter-notebooks benutzen.
2. Sie sind mit der grundlegenden python3 Funktionalität vertraut.

1.2 Aufträge

1. Installieren Sie (falls nicht schon lange geschehen) python3, numpy, scipy, matplotlib und jupyter.
2. Erarbeiten Sie das Einführungsbeispiel.
3. Vervollständigen Sie das vorliegende Jupyter-Notebook.

1.3 Aufgaben

1. Aufgabe

Erstellen Sie einen Graph der Funktion

$$f(x) = e^{-x^2/\sigma}$$

für $\sigma \in \{1/4, 1/3, 1/2\}$ inkl. Achsbeschriftung und Labels für $x \in [-2, 2]$.

```
[ ]:
```

2. Aufgabe

1. Programmieren Sie eine effiziente Funktion, welche die geometrische Folge

   $$x_n = \{q^k\}_{k=0}^n$$

   für ein gegebenes q und n berechnet. Definieren Sie den Parameter n mit dem Default-Wert 10.
2. Berechnen Sie das Skalarprodukt der beiden Vektoren gegeben durch die Folgen $\{0.5^k\}_{k=0}^{10}$ und $\{2^k\}_{k=0}^{10}$

```
[ ]:
```

3. Aufgabe

Aus der Analysis kennen Sie den Grenzwert der Folge

$$\lim_{k\to\infty} \left(1 + \frac{1}{2^k}\right)^{2^k} = e.$$

- Schreiben Sie ein Programm, welches die Folge berechnet und entscheiden, ob die numerische Berechnung korrekt ist.
  - Sie können mit Hilfe von `f = lambda x: x**2` inline Funktionen definieren.
- Wie gross kann k gewählt werden?

```
[ ]:
```

2 Grundlagen der Numerik

2.1 Lernziele

1. Sie kennen die unterschiedlichen Fehlerarten, welche in der Numerik zum Tragen kommen.
2. Sie kennen Differenzenquotienten zur Approximation von Ableitungen unterschiedlicher Ordnung.
3. Sie können die Fehlerordnung eines Differenzenquotienten experimentell und analytisch bestimmen.
4. Sie können die optimale Schrittweite eines Differenzenquotienten experimentell bestimmen.

2.2 Theorie

Differenzenquotienten zur Approximation von Ableitungen

Bei der numerischen Ableitung einer Funktion $f(x) = y$ an eienr Stelle $x_0 \in D_f$ wird der Grenzwert

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

durch die Sekantensteigung ersetzt

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

$h > 0$ nennen wir die Schrittweite.

Wir bestimmen die Fehlerordnung des mathematischen Verfahrens mit Hilfe der Taylorreihenentwicklung von f:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + \frac{f^{(3)}(x_0)}{6}h^3 + \mathcal{O}(h^4)$$

Durch Umformen dieser Gleichung erhalten wir

$$\frac{f(x_0 + h) - f(x_0)}{h} - f'(x_0) = \frac{f''(x_0)}{2}h + \frac{f^{(3)}(x_0)}{6}h^2 + \mathcal{O}(h^3) = \mathcal{O}(h)$$

Somit hat der Vorwärts-Differenzenquotient $\Delta^1_{h\rightarrow} = \frac{f(x_0 + h) - f(x_0)}{h}$ die Fehlerordnung 1, d.h. der Fehler hängt linear von der Schrittweite ab.

Weitere Differenzenquotienten für die Approximation der ersten Ableitung sind

- Rückwärts-Differenzenquotient: $\Delta^1_{h\leftarrow} = \frac{f(x_0) - f(x_0 - h)}{h}$ mit Fehlerordnung 1 und
- zentraler Differenzenquotient: $\Delta^1_{2h} = \frac{f(x_0 + h) - f(x_0 - h)}{2h}$ mit Fehlerordnung 2.

Das folgende Beispiel zeigt, dass neben dem Verfahrensfehler eine **weitere Fehlerart** zu berücksichtigen ist:

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Wir betrachten die Funktion

$$f(x) = \cos(x)$$

[2]:
```
def f(x):
    return np.cos(x)
```

im Punkt

$$x_0 = 1.$$

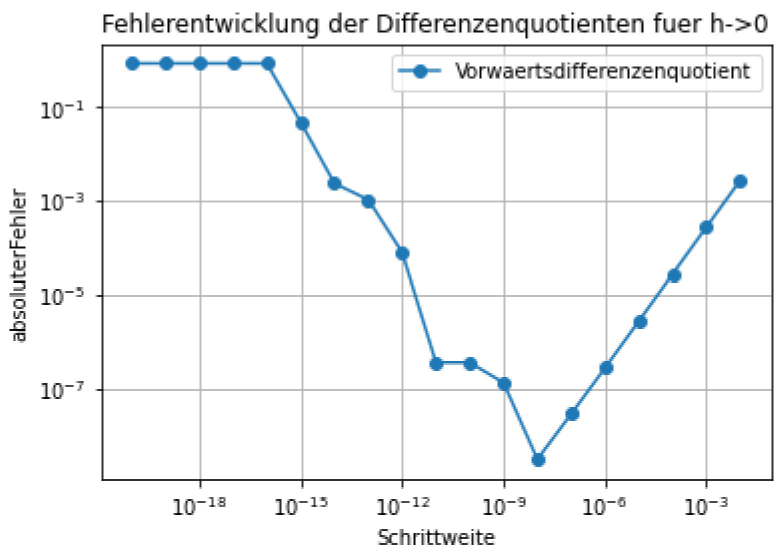
[3]:
```
x0 = 1
```

Wir berechnen nun die Differenzenquotienten mit unterschiedlichen Schrittweiten:

```
[4]: # exakter Wert der Ableitung
y = -np.sin(1)

DeltaRechts = []
DeltaLinks = []
DeltaZentral = []
Hs = 10.**np.arange(-20,-1) # logarithmische Schrittweite

for h in Hs:
    # Fehler des rechtsseitigen Differenzenquotient
    DeltaRechts.append(np.abs(y-(f(x0+h)-f(x0))/h))

    # Fehler des linksseitigen Differenzenquotient
    #DeltaLinks.append(<<snipp selber machen>>)

    # Fehler des zentralen Differenzenquotient
    #DeltaZentral.append(<<snipp selber machen>>)
```

Für die Analyse betrachten die Logarithmische Darstellung:

```
[6]: plt.loglog(Hs,DeltaRechts,'o-',label='Vorwaertsdifferenzenquotient')
# die beiden folgenden Zeilen können Sie nach dem Implementieren
# der weiteren Differenzenquotienten aktivieren
#plt.loglog(Hs,DeltaLinks,'.-',label='Rueckwaertsdifferenzenquotient')
#plt.loglog(Hs,DeltaZentral,'.-',label='zentraler DiffQuotient')
plt.xlabel('Schrittweite')
plt.ylabel('absoluterFehler')
plt.title('Fehlerentwicklung der Differenzenquotienten fuer h->0')
plt.legend()
plt.grid()
plt.show()
```



2.3 Aufträge

1. Analysieren Sie das Ergebnisse des obigen Beispiels. Welche Fehlerarten sind hier zu beobachten? Was folgern Sie daraus?
2. Implementieren Sie die Berechnung des Rueckwaerts- und zentralen Differenzenquotienten.
3. Leiten Sie mit Hilfe der Taylorreihe die Fehlerordnung für den zentralen Differenzenquotienten her (analog zur Einführung).
4. Effiziente Programmierung in Skript-Sprachen: Wie könnte man die obige for-Schleife vermeiden?

2.4 Abgabe

Bitte geben Sie Ihre Lösungen bis spätestens vor dem nächsten Praktikum 2 ab.

Downloads:

- PDF-Dokumentation:
  - `Anleitung Praktikum 1`
- Jupyter-Notebooks:
  - `Jupyter-Notebook Einführung in Python 3`
  - `Jupyter-Notebook Grundlagen der Numerik`