Semesterarbeit Teil 4a: Visualisierung von Taylor-Polynomen

Aufgabenstellung

Visualisieren Sie die Approximation einer Funktion durch Taylorpolynome (z.B. von sin(x) oder von ln(x+1).)

Schreibem Sie eine Python-Funktion, die die Graphen der Ursprungsfunktion sowie der Taylorpolynome bis zu einem bestimmten Grad mit Matplotlib visualisiert.

Implementation

Struktur

Für diese Arbeit verwende ich folgende Scripts:

- «test.py»: Dieses Script dient zur Überprüfung der Implementationen.
- «tools.py»: Dieses Script enthält die Funktion mit dem Newton-Verfahren.

Scripts

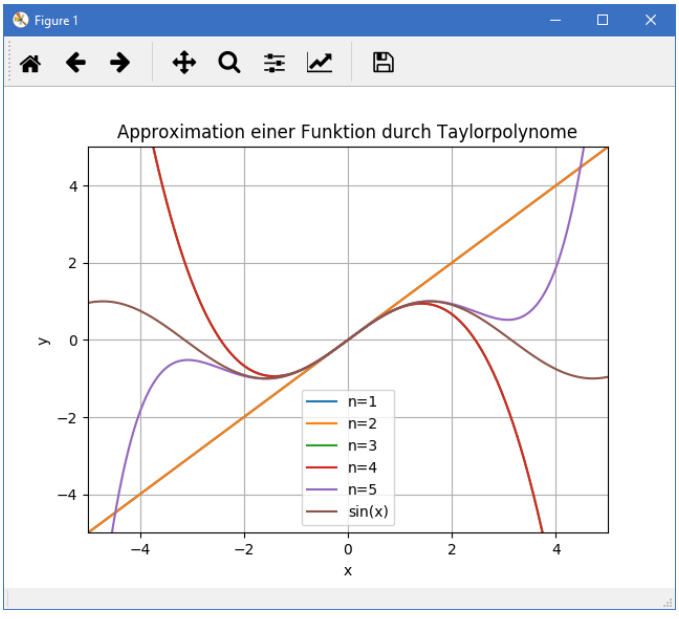
Für diese Semesterarbeit habe ich den Python-Code ausführlich kommentiert, damit ich hier mehr Platz für den Theorieteil der Aufgabe habe.

Resultat

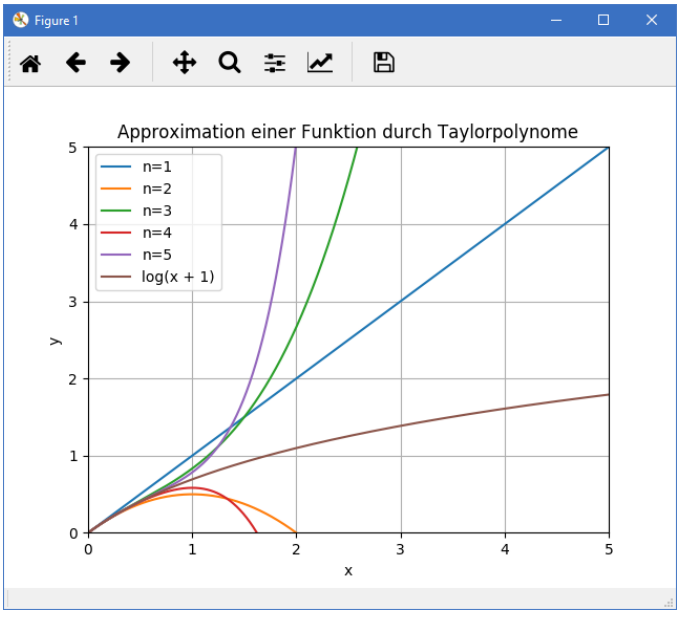
Zuerst hat man die Möglichkeit eine Funktion (aus den beiden Beispielfunktionen für die Aufgabe) zu wählen. Dann wird der entsprechende Graph durch die Funktion «show_taylor_graph(fnc, depth, grid_min, grid_max)» generiert:

```
Funktionen:
------------
`0`: sin(x)
`1`: log(x + 1)

Bitte wählen Sie einen Ausgabetypen: 1
```



Grafik 1: sin(x)



Grafik 2: ln(x+1)

Theorieteil

Mit der Taylorformel kann man das Taylorpolynom errechnen:

$$f(x) \approx f(x_e) + f'(x_e)(x-x_e) + \frac{f''(x_e)}{2!}(x-x_e)^2 + \frac{f'''(x_e)}{3!}(x-x_e)^3 + \ldots + \frac{f^{(n)}(x_e)}{n!}(x-x_e)^n$$

wenn $x \approx x_e$

$$f(x) \approx \sum_{k=0}^{n} \frac{f^{(k)}(x_e)}{k!}(x-x_e)^k$$

wenn $x \approx x_e$

Die Bilder mit der Formel habe ich kopiert von:

http://www.mathematik.net/reihen-taylorreihen/ty2s25.htm

Beide Formeln kommen auf dasselbe Resultat, die oben einfach ohne und die unten mit Summenzeichen.

Zu den Variablen:

- f(x): Ist das Taylorpolynom
- n: Ist der Grad des Taylorpolynoms
- x_e: Ist die Entwicklungstelle des Taylorpolynoms
- x: Unbekannte welche ungefähr x_e entspricht
- k: Wird bis nach n, also nach dem Grad hochgezählt für jede Summenzeicheninstanz
- $f^{(k)}$: Ist die k-te Ableitung von $f(x_e)$

Und wieso machen wir das alles?

- Durch die Taylorpolynome können Funktionen und einzelne Funktionswerte relativ exakt approximiert werden.
- Je grösser der Grad (also die Zahl n) ist, desto grösser ist die Genauigkeit der Approximation.
- Polynome sind auf eine leichte Art und Weise (beliebig oft) differenzierbar.
- Polynome sind verhältnismässig leicht zu integrieren.

Beispiel zu den Aussagen aus dem Python-Code: Wie man auf der Grafik 1 und auf der Grafik 2 klar erkennen kann, sind alle Taylor-Polynome anfangs noch sehr nahe an der Basisfunktion und weichen dann immer weiter ab. Je höher jedoch der Grad (also die Zahl n) ist, desto länger und näher befinden sich die Polynome auch an der Basisfunktion.

Das Programm gibt auch die Formeln für alle berechneten Polynome aus:

```
n=1 x
n=2 -x**2/2 + x
n=3 x**3/3 - x**2/2 + x
n=4 -x**4/4 + x**3/3 - x**2/2 + x
n=5 x**5/5 - x**4/4 + x**3/3 - x**2/2 + x
```

Hier kann man das gut das wiederholende Summenmuster erkennen (ich habe es farblich gekennzeichnet).

«k!» ist eine Fakultät: «Die Fakultät (manchmal, besonders in Österreich, auch Faktorielle genannt) ist in der Mathematik eine Funktion, die einer natürlichen Zahl das Produkt aller natürlichen Zahlen (ohne Null) kleiner und gleich dieser Zahl zuordnet.» - https://de.wikipedia.org/wiki/Fakult%C3%A4t_(Mathematik)