LEAN Buch

Release 0.1

Lea Soffel

Jun 09, 2023

CONTENTS

CHAPTER

ONE

EINLEITUNG

Dieses Buch thematisiert das Taylorpolynom. Es soll erst formal definiert werden und dann mit LEAN hergeleitet und beispielhaft berechnet werden. Die Schritte zur Erstellung des Taylorpolynoms sind aufgezeigt und werden an einem Beispiel ausprobiert. Der LEAN Code für das Beispiel kann mithilfe des *Try it!* Buttons in ein LEAN File kopiert und dort ausprobiert werden.

CHAPTER

TWO

DEFINITION

In diesem Kapitel wollen wir uns kurz die formale Defintion eines Taylorpolynoms anschauen.

Die Formel für das Taylorpolynom lautet:

$$T(x) = \sum_{k=0}^{n} \frac{f^(k)(x_0) \cdot (x - x_0)}{k!}$$

Das Taylorpolynom bildet die Summe über die Ableitungen bis Grad n am Entwicklungspunkt x_0 multipliziert mit $(x-x_0)$ und geteilt durch die Fakultät des Ableitungsgrades.

Um das Taylorpolynom zu bestimmen, müssen die folgenden Schritte abgehandelt werden:

1. Die ersten n Ableitung der Funktion f bestimmen.
2. Die Werte der Ableitungen an der Stelle x_0 berechnen.
3. Die Werte in die Formel einsetzen.

Im nachfolgenden Kapitel wollen wir diese Schritte an einem Beispiel ausführen.

CHAPTER

THREE

BEISPIEL

Wir wollen nun am Beispiel der Funktion

$$f(x) = 7x^4 + x^3 + 3x^2 + 5x - 7$$

das Taylorpolynon des 4. Grades aufstellen. Als Entwicklungspunkt wählen wir den Punkt $x_0 = 1$

Wir haben also gegeben

- $f(x) = 7x^4 + x^3 + 3x^2 + 5x - 7$,
- $x_0 = 1$ und
- $n = 4$.

Bevor wir mit dem ersten Schritt loslegen müssen wir zuerst die Funktion f implementieren.

```
def f (x: nat) := () -- dein Code hier
```

Um deine Funktion zu überprüfen kannst du die folgenden beiden Werte ausrechnen lassen:

- $f(0) = -7$
- $f(5) = 13768$

```
#eval f 0       -- -7
#eval f 5       -- 13768
```

Eine weitere Formel die wir aufstellen müssen, bevor wir loslegen können ist die Fakultätsfunktion. Zur Erinnerung es gibt 2 Fälle: - Fakultät von 0 ergibt 1, - Fakultät von n ergibt n * Fakultät von n-1

Kopiere den Codeschnipsel raus und vervollständige die Fakultätsfunktion. Mithilfe der *#eval* Statements kannst du deine Definition überprüfen.

```
def fac : ℕ → ℕ
| -- dein Code hier
| -- dein Code hier

-- Test
#eval fac 0     -- 1
#eval fac 5     -- 120
```

3.1 Ableitungen bestimmen

Im 1. Schritt müssen die Ableitungen definiert werden. Da wir das Taylorpolynom des 4. Grades berechnen wollen, benötigen wir die ersten 4 Ableitungen der Funktion $f(x)$.

Kopiere die unteren Codeschnipsel und füge jeweils die Ableitungen hinzu. Um dein Ergebnis zu testen, kannst du wieder die *#eval* Statements ausführen. Kommen die Ergebnisse, die als Kommentar dahinter notiert sind raus, so ist die Ableitung korrekt.

```
-- 1. Ableitungen definieren
def f' (x: nat) := () -- dein Code hier

-- Test
#eval f' 0      -- 5
#eval f' 5      -- 2735
```

```
-- 2. Ableitung definieren
def f'' (x: nat) := () -- dein Code hier

-- Test
#eval f'' 0     -- 6
#eval f'' 5     -- 1611
```

```
-- 3. Ableitung definieren
def f''' (x: nat) := () -- dein Code hier

-- Test
#eval f''' 0    -- 6
#eval f''' 5    -- 636
```

```
-- 4. Ableitung definieren
def f'''' (x: nat) := () -- dein Code hier

-- Test
#eval f'''' 0   -- 126
#eval f'''' 5   -- 126
```

3.2 Werte der Ableitungen am Entwicklungspunkt berechnen

Da LEAN die Werte selbst berechnen kann, können wir uns diesen Schritt sparen.

Wie in den *#eval* Statements schon gesehen, wird um eine Funktion/Definition mit einem Eingangsparameter zu berechnen, zuerst der Name der Funktion, dann ein Leerzeichen und dann die Zahl, die den Eingangsparameter darstellt geschrieben. Also z.B. *f' 0* bedeutet soviel wie *f'(0)*.

3.3 Werte in die Formel einsetzen

Zur Erinnerung, die Formel lautet:

$$T(x) = \sum_{k=0}^{n} \frac{f^(k)(x_0) \cdot (x - x_0)^1}{k!}$$

Angewendet auf unseren Anwendungsfall mit :math:'n = 4' ergibt sich:

$$T(x) = \sum_{k=0}^{4} \frac{f^(k)(1) \cdot (x - 1)^1}{k!}$$

Nun können wir mit den bereits definierten Ableitungen unsere Taylorpolynom-Formel aufstellen. :math:'x_0' geben wir vorerst als zweite Variable mit in unsere Taylorpolynom-Formel, so können wir für verschiedene Entwicklungspunkte das Taylorpolynom berechnen.

```
def taylor (x : nat) (x0 : nat) := () -- dein Code hier
```

Jetzt wollen wir unsere Formel auf den Entwicklungspunkt :math:'x_0 = 1' anwenden und unser Ergebnis testen:

```
#eval taylor 0 1 -- 9
#eval taylor 5 1 -- 3429
```

CHAPTER

FOUR

INDEX