MAT-INF1100 Oblig 2 - William Dugan

October 20, 2021

1 Oppgave 1

1.1 Oppgave 1a.

Vi skal bruke Taylor approksimasjon for å tilnærme funksjonen $f(x) = arctan(x)$ i punktet $a = 0$. Vi skal finne Taylorapproksimasjonen av 3. orden til f. Vi vet at $f(0) = 0$. Videre skal vi finne 1., 2., og 3. deriverte til f.

$$f'(x) = \frac{1}{1+x^2} => \frac{1}{1+0^2} * (x-0) = x$$

$$f''(x) = [(1+x^2)^{-1}]' = \frac{-2x}{(1+x^2)^2} => 0 * (x-0)^2 = 0$$

$$f'''(x) = \frac{(-2)(1+x^2)^2 - (-2x) * 2 * (1+x^2)(2x)}{(1+x^2)^4} = \frac{6x^2-2}{(1+x^2)^3} => -\frac{2}{3!} * (x-0)^3 = -\frac{x^3}{3}$$

Dermed er Taylorapproksimasjonen til f av 3. orden i punktet $a = 0$

$$T_3 f(x) = x - \frac{x^3}{3}. \tag{1}$$

1.2 Oppgave 1b.

I denne oppgaven skal vi plotte $f(x) = ln(x)$ og Taylorapproksimasjonene til f av orden 1, 2, og 3 i punktet $a = 1$. Jeg begynner med å finne uttrykk for de tre første leddene i Taylor rekken.

$$f(a) = 0$$

$$f'(x) = \frac{1}{x} => \frac{1}{1!} * (x-1) = (x-1)$$

$$f''(x) = -\frac{1}{x^2} => -\frac{1}{2!} * (x-1)^2 = -\frac{1}{2}(x-1)^2$$

$$f'''(x) = \frac{2}{x^3} => \frac{2}{3!} * (x-1)^3 = \frac{1}{3}(x-1)^3$$

Videre programmerer jeg dette i Python. Siden

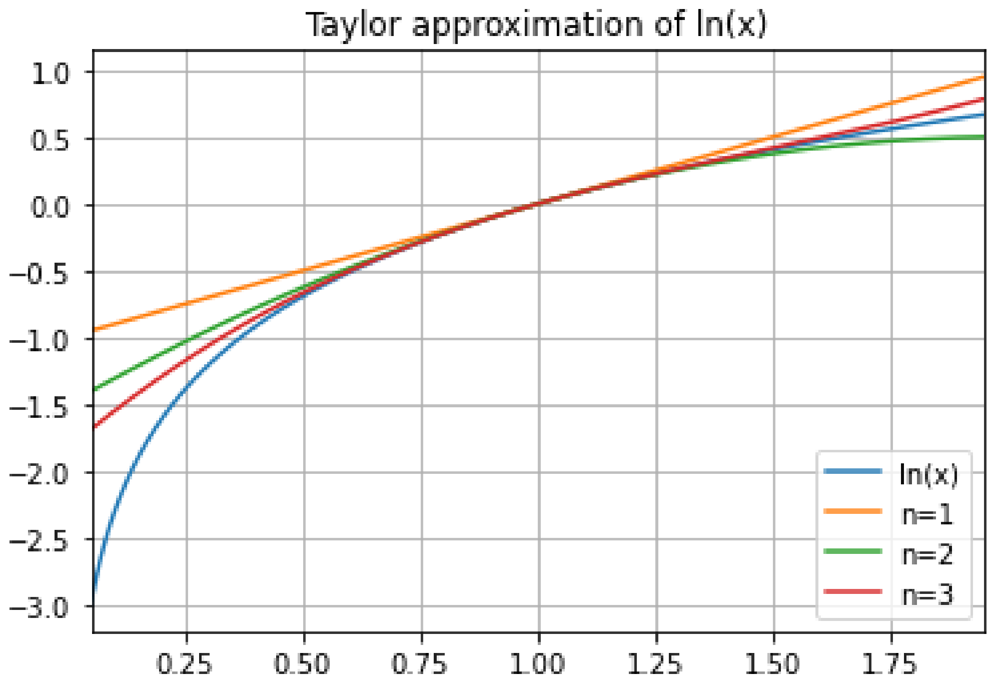
$$T_n f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{2}$$

har jeg definert $T_2f(x)$ som summen av $T_1f(x)$ og det det neste leddet i rekken. Koden ser slik ut:

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

def f(x):
    return np.log(x)

def T1f(x):
    return (x-1)

def T2f(x):
    return T1f(x) - 1/2 * (x-1) ** 2

def T3f(x):
    return T2f(x) + 1/3 * (x-1) ** 3
```

For å plotte dette bruker jeg matplotlib.pyplot med x-verdiene gitt i oppgaven.

[2]:
```python
x_min = 0.05
x_max = 1.95
points = 1000
x = np.linspace(x_min, x_max, points)

plt.plot(x, f(x), label='ln(x)')
plt.plot(x, T1f(x), label='n=1')
plt.plot(x, T2f(x), label='n=2')
plt.plot(x, T3f(x), label='n=3')

plt.title('Taylor approximation of ln(x)')
plt.xlim(x_min, x_max)
plt.legend(loc='lower right')
plt.grid()
plt.show()
```



Her ser vi at approksimasjonen til f blir bedre med flere ledd i Taylor rekken.

2 Oppgave 2

2.1 Oppgave 2a.

Vi er gitt funksjonen $f(x) = \frac{1}{x^2}$. Vi skal vise med induksjon at $f^{(k)}(x) = (-1)^k(k+1)!/x^{k+2}$ for alle $k \geq 0$. Vi begynner med å teste at påstanden stemmer for $n = 0$. Vi vet at $f^0(x) = f(x)$.

$$f^0(x) = \frac{(-1)^0(0+1)!}{x^{0+2}} = \frac{1}{x^2}$$

Videre antar vi at påstanden stemmer for $n = 0, 1, ..., k$. Vi må vise at påstanden stemmer for $n = k+1$. Vi har

$$\begin{aligned} f^k(x) &= \frac{(-1)^k(k+1)!}{x^{k+2}} \\ f^{k+1}(x) &= \frac{(-1)^{k+1}((k+1)+1)!}{x^{(k+1)+2}} \\ &= \frac{(-1)^{k+1}(k+2)!}{x^{k+3}} \end{aligned}$$

Siden $f^{k+1}(x) = D[f^k(x)]$ må vi vise at $\frac{d}{dx}f^k(x) = f^{k+1}(x)$.

$$
\begin{aligned}
\frac{d}{dx} f^k(x) &= \frac{(-1)^{k+1}(k+2)(k+1)!}{x^{k+3}} \\
&= \frac{(-1)^{k+1}(k+2)!}{x^{k+3}} \\
&= f^{k+1}(x)
\end{aligned}
$$

Vi har nå vist med induksjon at $f^{(k)}(x) = (-1)^k(k+1)!/x^{k+2}$ for alle $k \geq 0$.

2.2 Oppgave 2b.

Vi skal finne Taylorapproksimasjonen $T_n f(x)$ til f i punktet $a = 1$ og et uttrykk for restleddet $R_n f(x)$. Vi bruker (2) og formelen for den n-te deriverte til f og $a = 1$

$$
\begin{aligned}
T_n f(x) &= \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \\
&= \sum_{k=0}^{n} \frac{(-1)^k \frac{(k+1)!}{a^{k+2}}}{k!}(x-a)^k \\
&= \sum_{k=0}^{n} (-1)^k (k+1)(x-1)^k
\end{aligned}
$$

Videre bruker vi Lagranges restleddformel (11.2.3 i Kalkulus) for å finne et uttrykk for restleddet.

$$
\begin{aligned}
R_n f(x) &= \frac{f^{(n+1)(c)}}{(n+1)!}(x-a)^{n+1} \\
&= \frac{(-1)^{n+1} \frac{(n+2)!}{c^{n+3}}}{(n+1)!}(x-a)^{n+1} \\
&= (-1)^{n+1} \frac{n+2}{c^{n+3}}(x-1)^{n+1}
\end{aligned}
$$

2.3 Oppgave 2c.

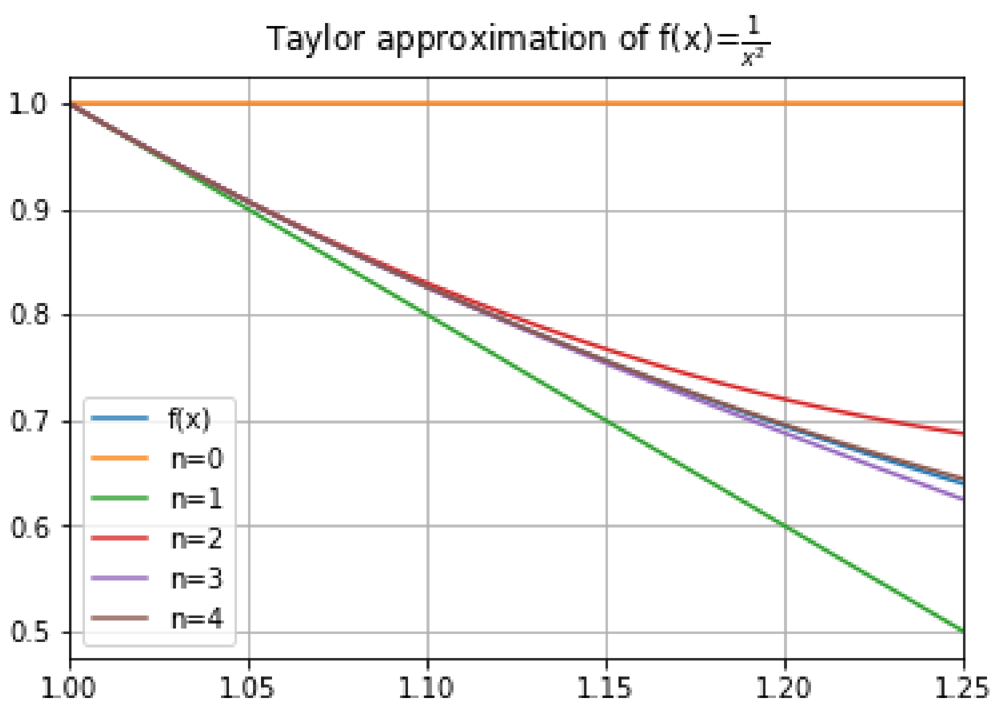
Jeg valgte å løse denne deloppgaven i Python, men det er helt sikkert mulig å løse den for hånd også. Jeg begynner med å definere uttrykkene fra oppg. 2b.

[3]:
```
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return 1 / x ** 2

def Tnf(x, n):
    s = 0
```

```
    for k in range(n+1):
        s += (-1)**k * (k+1) * (x-1)**k
    return s

def Rnf(x, c, n):
    return (-1)**(n+1) * ((n+2)/c**(n+3)) * (x-1)**(n+1)
```

Videre ønsker jeg å plotte Taylorapproksimasjonene for å sjekke at funksjonene er riktige og fungerer som forventet. Dette gjøres enkelt med matplotlib.pyplot.

[4]:
```
x = np.linspace(1, 1.25, 1000)
plt.plot(x, f(x), label='f(x)')

for n in range(5):
    plt.plot(x, Tnf(x, n), label=f'n={n}')

plt.title(r'Taylor approximation of f(x)=$\frac{1}{x^2}$')
plt.grid()
plt.xlim(1, 1.25)
plt.legend()
plt.show()
```



Jeg ser at tilnærmingen til f blir bedre med høyere grad av n. Videre skal vi finne et naturlig tall N slik at for alle $n \geq N$ og for alle $x \in [1, 1.25]$ skal feilen i $T_n f(x)$ være mindre enn 0.02 (altså $R_n f(x) \leq 0.02$). Fra grafen ovenfor ser jeg at feilen er størst i $x = 1.25$. Siden vi deler på c^{n+3} er feilen størst når $c = 1$. Dermed velger jeg å regne ut feilen for $x = 1.25, c = 1$ for å sørge for at $R_n f(x) \leq 0.02$ for alle $x \in [1, 1.25]$. Dette gjøres med en kort while-løkke.

[5]:
```
x = 1.25
c = 1.0
n = 0
error = Rnf(x, c, n)

while abs(error) > 0.02:
    n += 1
    error = Rnf(x, c, n)

print(error, n)
```

```
0.01953125 3
```

Programmet forteller at når $n = 3$ er feilen i $T_n f(x)$ som er ≤ 0.02. For å dobbeltsjekke at dette funker kjører jeg en kort testfunksjon som bruker at $R_n f(x) = f - T_n f(x)$. Denne sørger for at dersom det er tall $n \geq 3$ som gir feil større enn 0.02 stopper programmet og returnerer en feilmelding.

[6]:
```
def test_error():
    x = 1.25
    c = 1.0
    for n in range(1000, 3, -1):
        expected = f(x) - Tnf(x, n)
        tolerance = 0.02
        calculated = Rnf(x, c, n)
        sucess = abs(expected-calculated) < tolerance
        msg = f'Test failed at n = {n}'
        assert sucess, msg

test_error()
```

Koden skriver ikke ut noe som vil si at svaret vårt er riktig.