Matte 2 oblig

Martin Reistad

I dette prosjektet har jeg valgt å lage en graf for hvordan Chess.com ratingen min har utviklet seg de siste 350 partiene, for så å rekonstruere grafen så bra som mulig ved hjelp av fouriertransformasjon. Alt dette ble skrevet som python-kode. Det koden gjør er å implementere fouriertransformasjonen til et diskret signal ved å beregne fourierkoeffisientene og rekonstruere signalet med en gitt mengde ledd. Her er koden:

```python
import numpy as np
import matplotlib.pyplot as plt

#Del 1
y = [428, 445, 479, 504, 497, 465, 488, 483, 491, 491, 471, 442, 443, 454, 426, 421, 442, 454, 474, 479, 489, 518, 542, 532, 5
x = np.arange(0, len(y)*5, 5)

#Del 2
N = len(y)
ledd_hoy, ledd_lav = 25, 5
a_0 = np.mean(y)

#Del 3
def fourier_koeffisienter(ledd):
    a, b = np.zeros(ledd), np.zeros(ledd)
    for n in range(ledd):
        a[n] = (2 / N) * np.sum(y * np.cos(2 * np.pi * n * np.arange(N) / N))
        b[n] = (2 / N) * np.sum(y * np.sin(2 * np.pi * n * np.arange(N) / N))
    return a, b

#Del 4
def rekonstruer_signal(a, b, ledd):
    y_rekonstruert = np.zeros_like(y, dtype=float)
    for n in range(1, ledd):
        y_rekonstruert += a[n] * np.cos(2 * np.pi * n * np.arange(N) / N) + b[n] * np.sin(2 * np.pi * n * np.arange(N) / N)
    return y_rekonstruert + a_0

a_hoy, b_hoy = fourier_koeffisienter(ledd_hoy)
a_lav, b_lav = fourier_koeffisienter(ledd_lav)

rekonstruert_y_hoy = rekonstruer_signal(a_hoy, b_hoy, ledd_hoy)
rekonstruert_y_lav = rekonstruer_signal(a_lav, b_lav, ledd_lav)

#Del 5
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
plt.plot(x, y, label="Original Data")
plt.plot(x, rekonstruert_y_hoy, label=f"Fourier med {ledd_hoy} ledd", linestyle="--")
plt.xlabel('Parti')
plt.ylabel('Ranking')
plt.legend()
plt.title(f"Original Data vs Fourier med {ledd_hoy} ledd")

plt.subplot(2, 1, 2)
plt.plot(x, y, label="Original Data")
plt.plot(x, rekonstruert_y_lav, label=f"Fourier med {ledd_lav} ledd", linestyle="--")
plt.xlabel('Parti')
plt.ylabel('Ranking')
plt.legend()
plt.title(f"Original Data vs Fourier med {ledd_lav} ledd")

plt.tight_layout()
plt.show()
```

Koden er delt inn i 5 deler.

Del 1: Her lages liste over ranking og antall partier.

Del 2: Her lagres antall datapunkter til rankingen, ledd i fourierrekken og gjennomsnittlig verdi som variabler.

Del 3: Her beregnes fourierkoeffisientene ved hjelp av følgende formler:

$$a_n = \frac{2}{N} \sum_{n=0}^{N-1} y_n \cos\left(\frac{2\pi n k}{N}\right)$$

$$b_n = \frac{2}{N} \sum_{n=0}^{N-1} y_n \sin\left(\frac{2\pi n k}{N}\right)$$

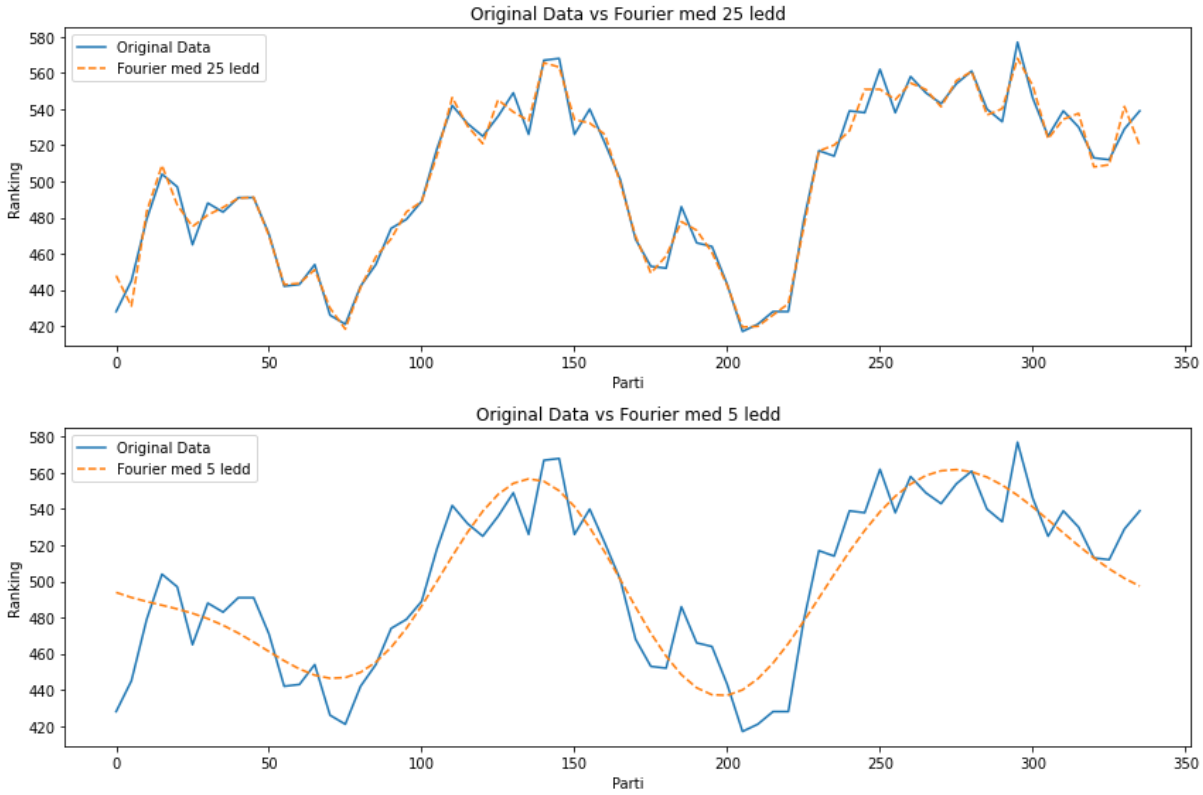
Istedenfor å bruke integraler brukes summer som summerer sammen arealet under garfen. Funksjonen i del 3 lager en liste med lengden til antall ledd i fourierrekken med verdiene for koeffisientene.

Del 4: Her rekonstrueres dataen ved hjelp av denne formelen:

$$f(t) = a_0 + \sum_{n=1}^{N-1} \left(a_n \cos\left(2\pi n t\right) + b_n \sin\left(2\pi n t\right)\right)$$

Del 5: Her plottes dataen.

Resultat:



Jeg testet med ulike ledd i fourierrekken, og kom fram til at 25 ledd ga en ønsket oppførsel. Under ser man hvordan det ser ut med 5 ledd. Dette viser at man vil kunne få en mer nøyaktig rekonstruksjon jo fler ledd man bruker i fourierrekken. Etter gjennomført prosjekt kan det konkluderes med at fouriertransformasjonen fungerte bra, og jeg ikke kommer til å bli en god sjakkspiller på veldig lang tid (hvertfall om jeg følger samme utvikling).