

# **OBLIGATORISK OPPGAVE 2**

---

## **STA-2003-Tidsrekker**

**22. mars 2019**

Martin Soria Røvang  
Universitetet i Tromsø

Inneholder 12 sider, inkludert forside.

## **Innhold**

<b>1</b>	<b>Oppgave</b>	<b>3</b>
1.1	a . . . . .	3
1.2	b . . . . .	4
1.3	c . . . . .	4
1.4	d . . . . .	5
<b>2</b>	<b>2</b>	<b>7</b>
2.1	a-b . . . . .	7
<b>3</b>	<b>Appendix</b>	<b>9</b>
<b>4</b>	<b>Referanser</b>	<b>12</b>

# 1 Oppgave

## 1.1 a

Har prosessen,

$$x(t) = x_1(t) + x_2(t) \quad (1)$$

,disse er *ukorrelererte* og *stasjonære* prosesser med forventning  $E[x_1] = E[x_2] = 0$ .

Vi har Wiener-Khinchin theoremene,

$$S_{xx}(f) = \mathcal{F}\{R_{xx}(\tau)\} = \int_{-\infty}^{\infty} R_{xx}(\tau) e^{-i2\pi f\tau} d\tau \quad (2)$$

$$R_{xx}(\tau) = \mathcal{F}^{-1}\{S_{xx}(f)\} = \int_{-\infty}^{\infty} S_{xx}(f) e^{i2\pi f\tau} df \quad (3)$$

Her er  $\mathcal{F}$  Fourier transformasjon. Bruker at,

$$R_{xx}(\tau) = E[x_{t+\tau} x_t] \quad (4)$$

Løser vi denne med tanke på ligning (1) får vi,

$$R_{xx}(\tau) = E[(x_1(t+\tau) + x_2(t+\tau))(x_1(t) + x_2(t))] \quad (5)$$

Som gir,

$$R_{xx}(\tau) = E[x_1(t+\tau)x_1(t)] + E[x_1(t+\tau)x_2(t)] + E[x_1(t)x_2(t)] + E[x_2(t+\tau)x_2(t)] \quad (6)$$

Siden prosessene er ukorrelererte vil man kunne separere forventningene,  $E[x_1(t+\tau)]E[x_2(t)]$

Dette gir da,

$$R_{xx}(\tau) = E[x_1(t+\tau)x_1(t)] + E[x_1(t+\tau)]E[x_2(t)] + E[x_1(t)]E[x_2(t+\tau)] + E[x_2(t+\tau)x_2(t)] \quad (7)$$

Dette kan separeres slik at,

$$R_{xx}(\tau) = E[x_1(t+\tau)x_1(t)] + E[x_1(t+\tau)]E[x_2(t)] + E[x_1(t)]E[x_2(t+\tau)] + E[x_2(t+\tau)x_2(t)] \quad (8)$$

Siden  $R_{x_1x_1}(\tau) = E[x_1(t+\tau)x_1(t)]$  får vi,

$$R_{xx}(\tau) = R_{x_1x_1}(\tau) + E[x_1(t+\tau)]E[x_2(t)] + E[x_1(t)]E[x_2(t+\tau)] + R_{x_2x_2}(\tau) \quad (9)$$

Vet at forventningen til prosessene er null,  $E[x_1(t+\tau)]E[x_2(t)] = 0$  og  $E[x_1(t)]E[x_2(t+\tau)] = 0$ ,

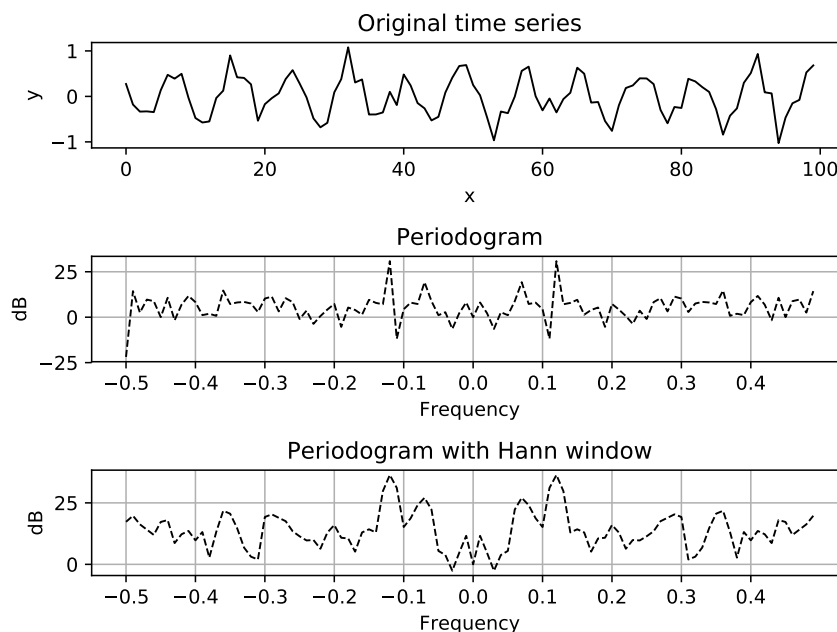
$$R_{xx}(\tau) = R_{x_1x_1}(\tau) + R_{x_2x_2}(\tau) \quad (10)$$

Siden vi har summer kan vi dele integralet opp i sum,

$$S_{xx}(f) = \underbrace{\int_{-\infty}^{\infty} R_{x_1x_1}(\tau) e^{-i2\pi f\tau} d\tau}_{=S_{x_1x_1}(f)} + \underbrace{\int_{-\infty}^{\infty} R_{x_2x_2}(\tau) e^{-i2\pi f\tau} d\tau}_{=S_{x_2x_2}(f)} \quad (11)$$

## 1.2 b

I figur(1) ser vi tidsrekken i øverste vindu, periodogram, og periodogram med hann vindu.



**Figur 1:** Beregnet periodogram av tidsrekken.

Her ser man at hann vinduet gir en glattere kurve, og lar ikke frekvenser lekke over like mye som i det vanlige periodogrammet. Frekvensene ser ut til å være rundt 0.12 og 0.07.

## 1.3 c

Som nevnt i oppgave b så fjerner vinduet mer av lekkasjen i frekvensspekteret, fordi den har mindre sideløber enn den idelle  $w[k] = 1$ , men dette vil øke bredden på hovedloben slik at man får litt dårligere oppløsning. I figure(2) har jeg ikke brukt dB skala slik at differansen er større mellom styrken på frekvensene. Her kan man tydelig se at vinduet hjelper med å se svakere periodisitet i spectrumet. Periodogram ble laget ved bruk av Fourier transform som vist i ligning(12).

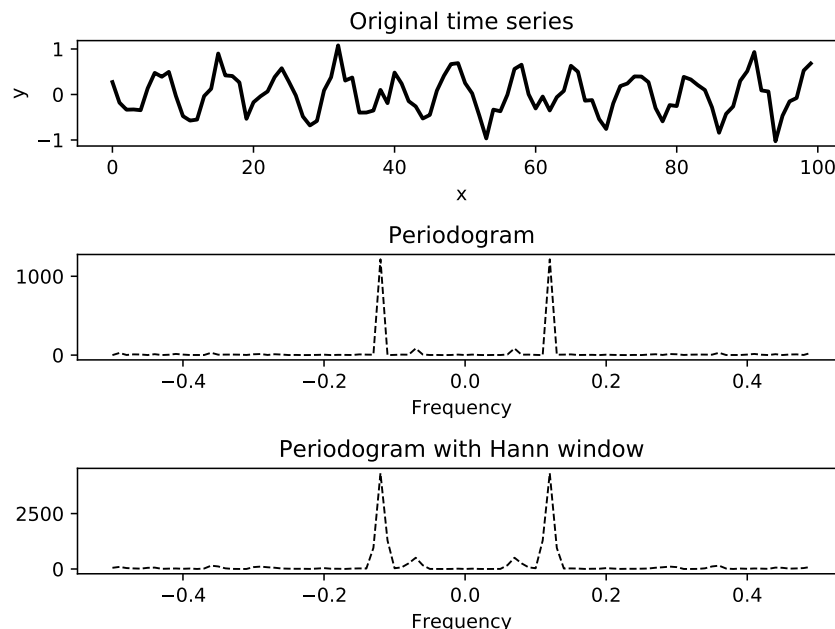
$$S(f) = \frac{1}{N} |\mathcal{F}\{x[n]\}|^2 \quad (12)$$

der  $\mathcal{F}$  er DTFT, *Discrete Time Fourier Transform* og N er antall datapunkter. Ved bruk av vindu har vi ligning(13).

$$S_w(f) = \frac{1}{NU} |\mathcal{F}\{w[n]x[n]\}|^2 \quad (13)$$

Her er  $w[n]$  vinduet og U er normaliseringskonstant for å fjerne energien som blir lagt til prosessen fra vinduet, i dette tilfelle er  $w[n]$  *Hann window*. Vi bruker vindu for å minske spektral lekkasje som kommer av at vi ikke kan ha en *ekte* fourier transform som krever at signal vi har er uendelig langt. Derfor vil vi alltid ha en ende på signal som man antar er null. Dette betyr at vi bruker et ideelt vindu med lengde M(signal lengden) når vi tar Fourier transform. Når man ganger inn tids domenet så konvolverer

man i frekvensdomenet, og siden det ideelle vinduet har store sideløber medfører dette til at man får ut nærliggende frekvenser i hverandre og dermed gir spektral lekkasje. Ved bruk av vindu som Hann vindu så kan man redusere dette siden de har smalere sideløber.

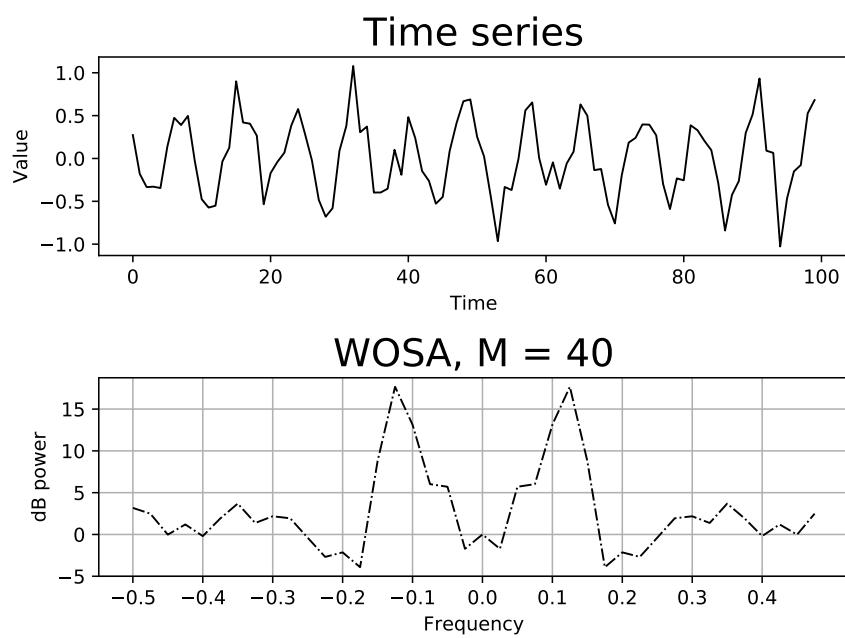


**Figur 2:** Beregnet periodogram av tidsrekken uten dB skala.

## 1.4 d

Ved å implementere WOSA (Weighted/Welchs overlapped segment averaging) på tidsrekken fikk vi resultatet vist i figur(3).

Her ble hvert segment vektet som gir et kraftigere utslag der det er sterke periodisiteter i forhold til resten av spekteret. Her ser man at frekvensen på ca  $f \approx 0.15$  og  $\approx 0.05$  er ganske kraftige i forhold til alle de andre. Man kan se at frekvensene er mye tydeligere i WOSA enn i periodogrammet og hann vinduet i dB-skala som vist i figur(1).

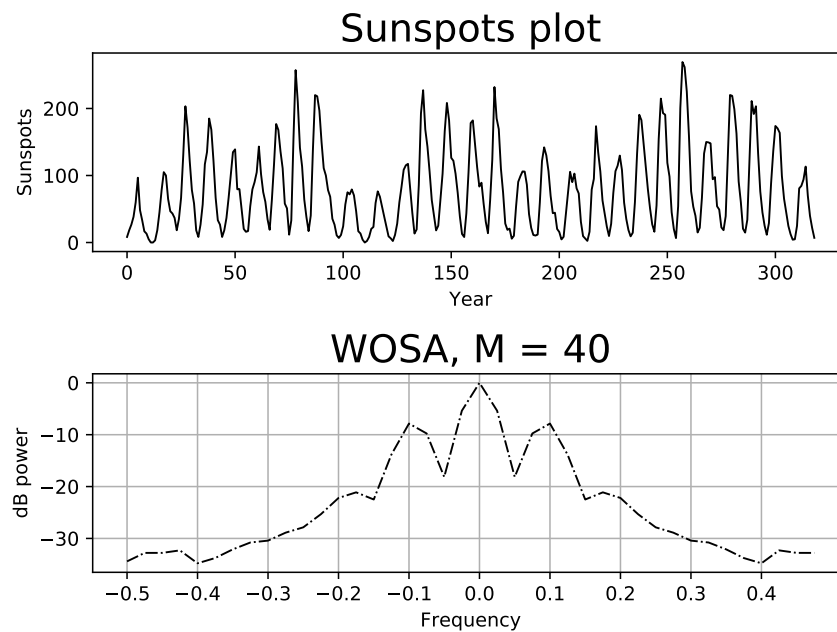


**Figur 3:** WOSA med 50% overlapp,  $N = 40$

## 2 2

### 2.1 a-b

Her plottet vi sol-dataen som vist i øverste vindu i figur(4).



**Figur 4:** Soldata, WOSA med 50% overlapp,  $N = 40$

Her ble det brukt WOSA med 50% overlapp og vindu-størrelse  $N = 40$ . Her ser man sterk  $DC(f = 0)$  og en frekvens på rundt 0.1,  $f \approx 0.1$ . Dette gir en periode på  $T \approx 10$ , noe som ser ut til å stemme hvis man teller antall topper som er  $\approx 10$  per 100.





### 3 Appendix

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4
5 data = np.genfromtxt('tidsrekke_oblig2_oppg1.txt')
6
7
8 def periodogram(x, dt):
9     """Regular periodogram"""
10    N = len(x)
11    spectrum = np.abs(np.fft.fftshift(np.fft.fft(x)**2))
12    spectrum *= dt/ N
13    freq = np.fft.fftshift(np.fft.fftfreq(N, dt))
14
15    return freq, spectrum
16
17
18 def w_periodogram(x, dt):
19     """Windowed periodogram"""
20    N = len(x)
21    n = np.arange(0,N,1)
22    # Hann window
23    window = (1/2)*(1 - np.cos(2*np.pi*n/(N-1)))
24    U = (1/N)*np.sum(window**2)
25    spectrum = np.abs(np.fft.fftshift(np.fft.fft(window*x)))**2
26    spectrum *= (dt/(N*U))
27    freq = np.fft.fftshift(np.fft.fftfreq(N, dt))
28    return freq, spectrum
29
30 dt = 1
31 freq, spectrum = periodogram(data, dt)
32 freqw, wspectrum = w_periodogram(data, dt)
33
34 # Find index corresponding to f = 0
35 idx = np.where(freq == 0)
36 widx = np.where(freqw == 0)
37
38 # Plot
39 fig, ax = plt.subplots(3,1)
40 ax[0].plot(data, color = 'black', linewidth = 1)
41 ax[0].set_title('Original time series')
42 ax[0].set_xlabel('x')
43 ax[0].set_ylabel('y')
44 ax[1].plot(freq, 10*np.log10(spectrum/spectrum[idx]), '--', color = 'black',
45 linewidth = 1)
46 ax[1].set_title('Periodogram')
47 ax[1].set_xlabel('Frequency')
48 ax[1].set_ylabel('dB')
49 ax[2].plot(freqw, 10*np.log10(wspectrum/wspectrum[widx]), '--', color = 'black',
50 linewidth = 1)
51 ax[2].set_title('Periodogram with Hann window')
52 ax[2].set_xlabel('Frequency')
53 ax[2].set_ylabel('dB')
54 ax[1].set_xticks([x for x in np.arange(-0.5,0.5,0.1)])
55 ax[2].set_xticks([x for x in np.arange(-0.5,0.5,0.1)])
56 ax[1].grid()
57 ax[2].grid()
58 plt.tight_layout()
59 plt.savefig('rapport/taskb.pdf', bbox_inches = 'tight',
60 pad_inches = 0)
61 plt.show()

```

**Figur 5:** Task 1B-C

```

1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  # Load file
5  data = np.genfromtxt('tidsrekke_oblig2_oppg1.txt')
6
7
8  def WOSA(x, M, dt = 1):
9      """Implementation of WOSA"""
10     n = np.arange(0, M, 1)
11     window = (1/2)*(1 - np.cos(2*np.pi*n/(M-1)))
12     U = (1/M)*np.sum(window**2)
13     spectrum = np.zeros(M)
14     n_windows = 2*int(len(x)/(M-1))
15     for i in range(n_windows):
16
17         # Start with window 0-40
18         if i == 0:
19             spectrum_temp = np.fft.fftshift(np.fft.fft(window*x[0:40]))
20             plt.plot(x[0:40])
21             t = np.arange(0, 40, 1)
22             plt.plot(t, window)
23
24
25         # Start overlapping
26         else:
27             spectrum_temp = np.fft.fftshift(np.fft.fft(window*x[(i*int(M/2)):(i
+2)*int(M/2)]))
28             plt.plot(x[(i+2)*int(M/2)])
29             t = np.arange((i*int(M/2)), (i+2)*int(M/2), 1)
30             plt.plot(t, window)
31             spectrum += (dt/(M*U))*np.abs(spectrum_temp)**2
32     spectrum /= n_windows
33     plt.show()
34     freq = np.fft.fftshift(np.fft.fftfreq(M, dt))
35     print('Number of windows: %s'%n_windows)
36     return freq, spectrum
37
38 M = 40
39 dt = 1
40 freq, spectrum = WOSA(data, M)
41 # Find index corresponding to f = 0
42 idx = np.where(freq == 0)
43
44 # Plot
45 fig, ax = plt.subplots(2,1)
46 ax[0].plot(data, linewidth = '1', color = 'black')
47 ax[0].set_title('Time series', fontsize = '20')
48 ax[0].set_ylabel('Value')
49 ax[0].set_xlabel('Time')
50 ax[1].plot(freq, 10*np.log10(spectrum/spectrum[idx]), '-', linewidth = '1',
color = 'black')
51 ax[1].set_title('WOSA, M = %s'%M, fontsize = '20')
52 ax[1].set_xlabel('Frequency')
53 ax[1].set_ylabel('dB power')
54 ax[1].set_xticks([x for x in np.arange(-0.5,0.5,0.1)])
55 ax[1].grid()
56 plt.tight_layout()
57 plt.savefig('rapport/taskd.pdf', bbox_inches = 'tight',
pad_inches = 0)
58
59 plt.show()
60

```

Figur 6: Task D

```

1 import os
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 filedir = os.path.dirname(__file__)
7 filename = 'SN_y_tot_V2.0.txt'
8
9 file = os.path.join(filedir, filename)
10
11 # load file
12 datatable = pd.read_csv(file, sep='\t', header=None, engine='python')
13 time = np.zeros(len(datatable))
14 sunspots = np.zeros(len(datatable))
15
16 for i in range(len(datatable)):
17     time[i] = datatable.values[i,0][0:6]
18     sunspots[i] = datatable.values[i,0][8:13]
19
20
21 def WOSA(x, M, dt = 1):
22     """Implementation of WOSA"""
23     n = np.arange(0, M, 1)
24     window = (1/2)*(1 - np.cos(2*np.pi*n/(M-1)))
25     U = (1/M)*np.sum(window**2)
26     spectrum = np.zeros(M)
27     n_windows = 2*int(len(x)/(M-1))-1
28     x = np.pad(x, (0, M), 'constant')
29     for i in range(n_windows):
30         # Start with window 0-40
31         if i == 0:
32             spectrum_temp = np.fft.fftshift(np.fft.fft(window*x[0:40]))
33             plt.plot(x[0:40])
34             t = np.arange(0, 40, 1)
35             plt.plot(t, 200*window)
36
37         # Start overlapping
38         else:
39             spectrum_temp = np.fft.fftshift(np.fft.fft(window*x[(i*int(M/2)):(i
40 +2)*int(M/2)]))
41             plt.plot(x[0:(i+2)*int(M/2)])
42             t = np.arange((i*int(M/2)), (i+2)*int(M/2), 1)
43             plt.plot(t, 200*window)
44             spectrum += (dt/(M*U))*np.abs(spectrum_temp)**2
45         spectrum /= n_windows
46         plt.show()
47         plt.close()
48         freq = np.fft.fftshift(np.fft.fftfreq(M, dt))
49         print('Number of windows: %s'%n_windows)
50         return freq, spectrum
51
52 M = 40
53 dt = 1
54 freq, spectrum = WOSA(sunspots, M)
55 # Find index corresponding to f = 0
56 idx = np.where(freq == 0)
57
58 # plot
59 fig, ax = plt.subplots(2,1)
60 ax[0].plot(sunspots, linewidth = '3', color = 'black')
61 ax[0].set_title('Sunspots plot', fontsize = '20')
62 ax[0].set_ylabel('Sunspots')
63 ax[0].set_xlabel('Year')
64 ax[1].plot(freq, 10*np.log10(spectrum/spectrum[idx]), '--', linewidth = '1',
65 color = 'black')
66 ax[1].set_title('WOSA, M = %s'%M, fontsize = '20')
67 ax[1].set_xlabel('Frequency')
68 ax[1].set_ylabel('dB power')
69 ax[1].set_xticks([x for x in np.arange(-0.5,0.5,0.1)])
70 ax[1].grid()
71 plt.tight_layout()
72 plt.savefig('rapport/task2.pdf', bbox_inches = 'tight',
73 pad_inches = 0)
74 plt.show()

```

## **4 Referanser**