FYS2130 Oblig fra Regneoppgaver 6

Domantas Sakalys

April 13, 2021

Oppgave 1: Aliasing

Fra oppgaveteksten har vi fått funksjonen

$$g(t) = 1\sin(2\pi f t)$$

Deloppgave a)

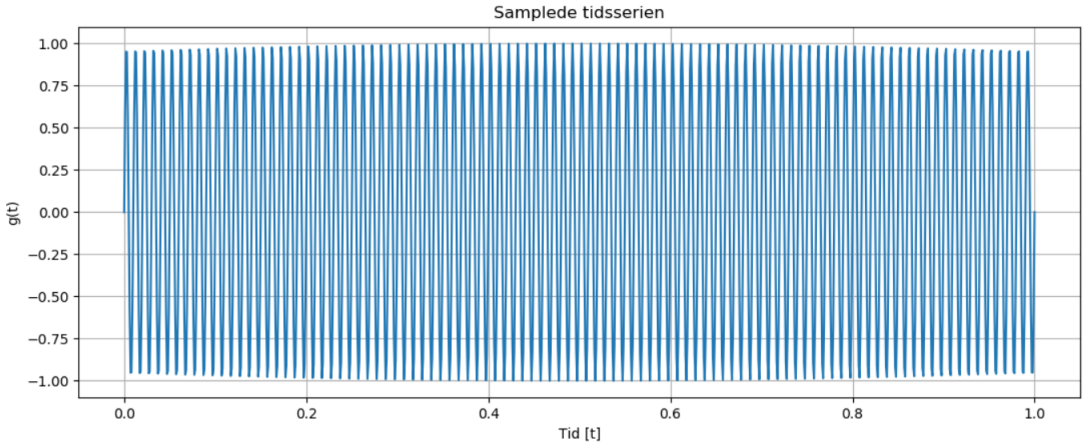
Vi starter med å gjøre frekvensanalyse av et signal med $f = 100$Hz, og samplingsfrekvens på $f_s = 1000$Hz.

Vi velger samplingstiden til å være 1 sekund. Utifra det, kan vi plotte den samplede tidserien. Det vil si at vi ønsker å plotte den diskretiserte $g(t)$, med spesifik samplingsfrekvens på 1kHz.
For å gjøre det, kan det være nyttig å vite akkurat antall samplinger vi ønsker å ha i denne samplingstiden. Antall samplinger er delt etter Δt, som er lik $1/f_s$ hvor f_s er samplingsfrekvensen. Utifra dette, finner vi antall samplinger,

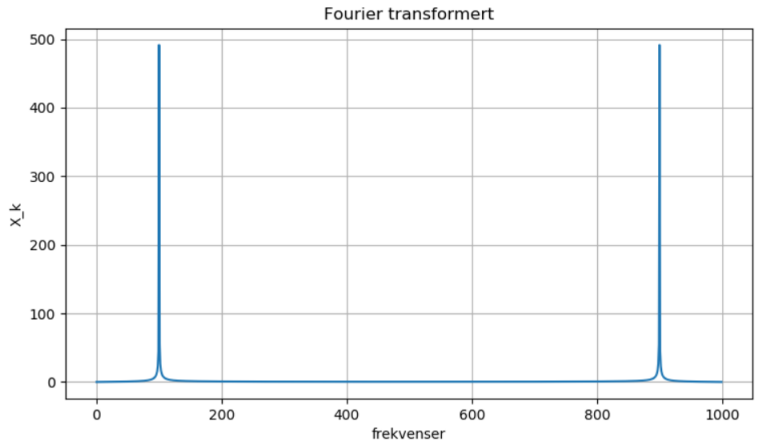
$$N = Tf_s = 1000$$

Det vil si at vi diskretiserer $g(t)$ inn i 1000 punkter, når t går fra 0 til 1 sekund.

Ved å plotte dette får vi dette som resultat,



Hvis vi nå velger å plotte dens diskrete fourierstransform, får vi plot av X_k som resultat. X_k er altså funksjonen til fouriertransform fra x_n.

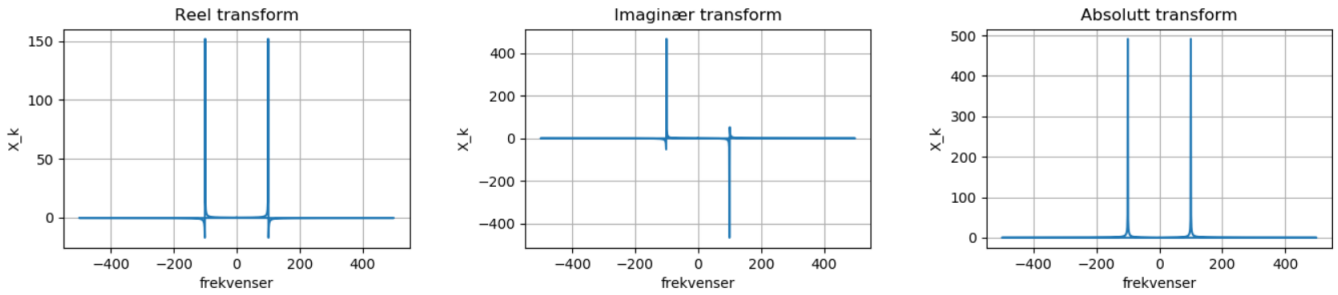


I koden some er lagt som vedlegg bakers i denne pdf'en, har jeg plottet absoluttverdien til X_k. Vi ville ha fått lignende resultat hvis vi hadde plottet realdelen av X_k.

Det er en ting man kan merke seg her, vi får 2 signaler! Vi vet at frekvensen til vår signal må være 100 Hz, men ved fouriertransform får vi vite at den er også på 900 Hz. Dette er på grunn av den absoluttverdiens signal konjugat symmetri. Det vil si at når det testes forskjellige verdier av frekvens i fouriertransfor, så får vi også et utslag når det intreffer komplekse negative verdier. Her er jeg litt usikker på hva dette egentlig betyr.

For å fikse dette problemet og eventuell misforståelse for hvilke frekvens vår

signal virkelig har, bruker vi innebygd python triks som er oppgitt i kodevedlegget. Ved å plotte den igjen, får vi dette plot som resultat.



Her har vi plottet Reel-, imaginør og absolutt transformasjon. Dermed kan vi se at signalet vårt har frekvens med 100Hz.
For å sjekke om det stemmer med med amplituden, må vi forstå hva som virkelig foregår når vi fourier transformerer. Utifra teorien, vet vi at når vi fourier transformerer, utfører vi denne integralet

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} d\omega$$

Hvor $f(x)$ er vår funksjon som vi ønsker å transformere, i vår tilfelle, er dette $g(t)$. Det er her vi kan tilpasse den andre faktoren i vårt integral. Vi kan velge å multiplisere med reel delen av det komplekse faktoren, eller komplekse delen. Vi kan også velge absoluttdelen. Dette e representert i de tre plotten øverst. Hva vi velger, kan avhenge av $f(x)$.

Siden $g(t)$ er en sinus funksjon, kan vi godt velge å det komplekse delen av den faktoren. I tillegg, kan vi velge å integrere fra 0 til 1000 Hz. Integralet vårt blir dermed,

$$\frac{1}{T} \int_0^{1000} \sin(2\pi f t) \sin(2\pi f t) dt$$

Ved å velge f til å være 100, ender vi opp med løsningen til å være på -500. Hvis vi ser på plottet øverst der hvor vi har plottet imaginær transform, ser vi at dette stemmer nesten! Korrekt amplitude fra plottet er på ca. -467.
Jeg tenker at feilen kan skilles fra samplingsrate som vi har brukt.

Hva skjer hvis vi bruker høyrere signalfrekvens?

Vi skal nå prøve å bruke høyrere signalfrekvens, samtidig beholde samme samplingsfrekvens på 1kHz. Vi skal prøve ut signalfrekvens på 400Hz, 700Hz og 1300Hz.

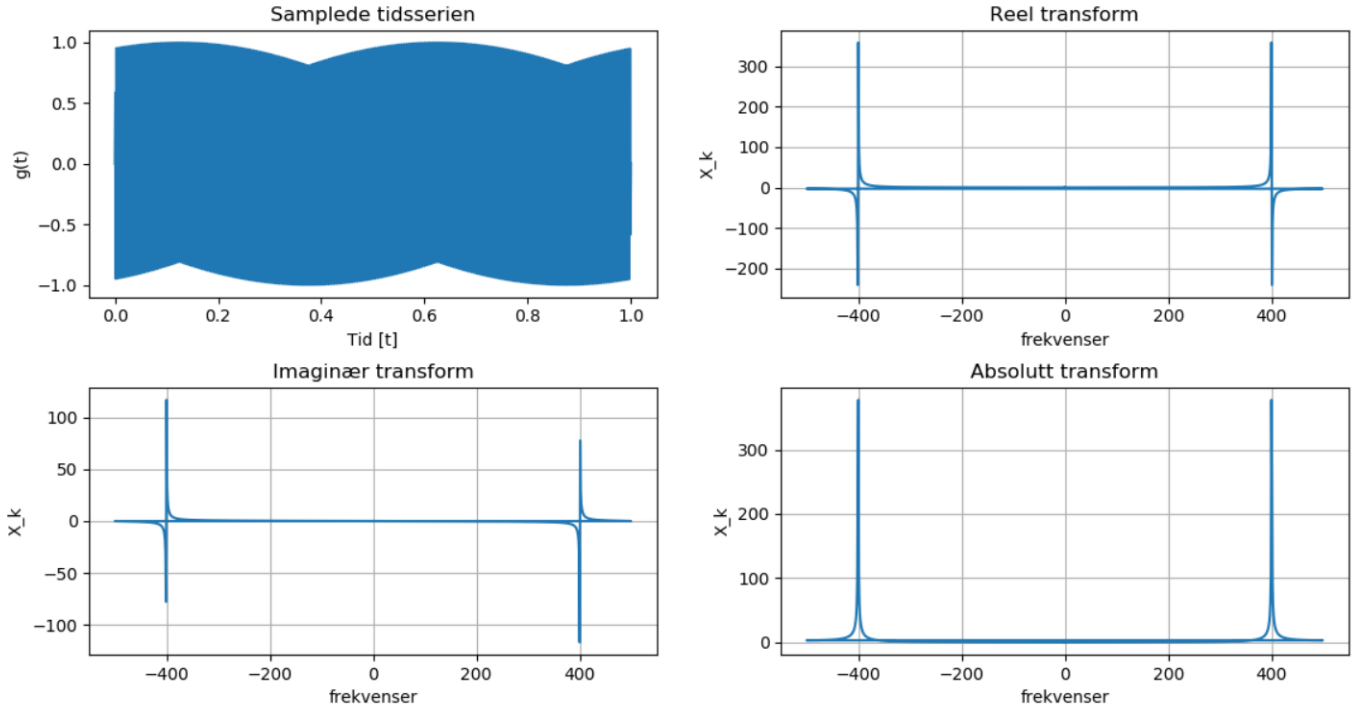


Figure 1: Plot med signalfrekvens på 400Hz

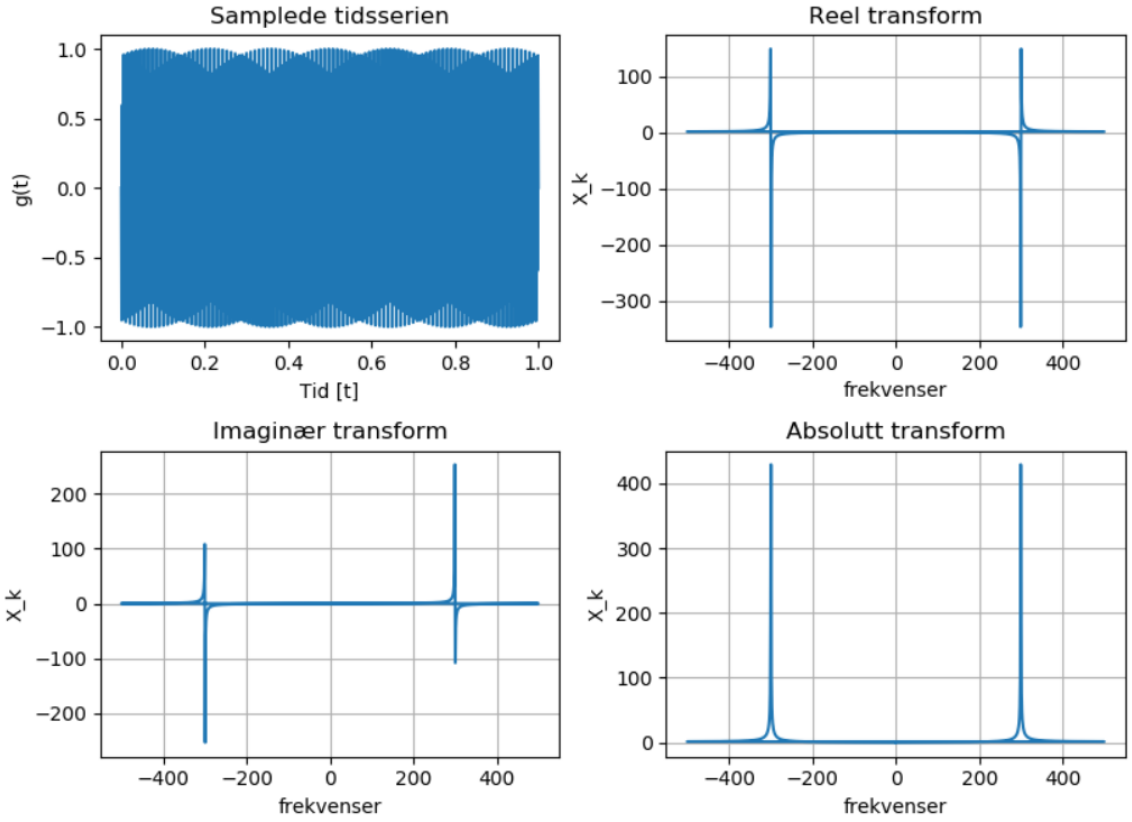


Figure 2: Plot med signalfrekvens på 700Hz

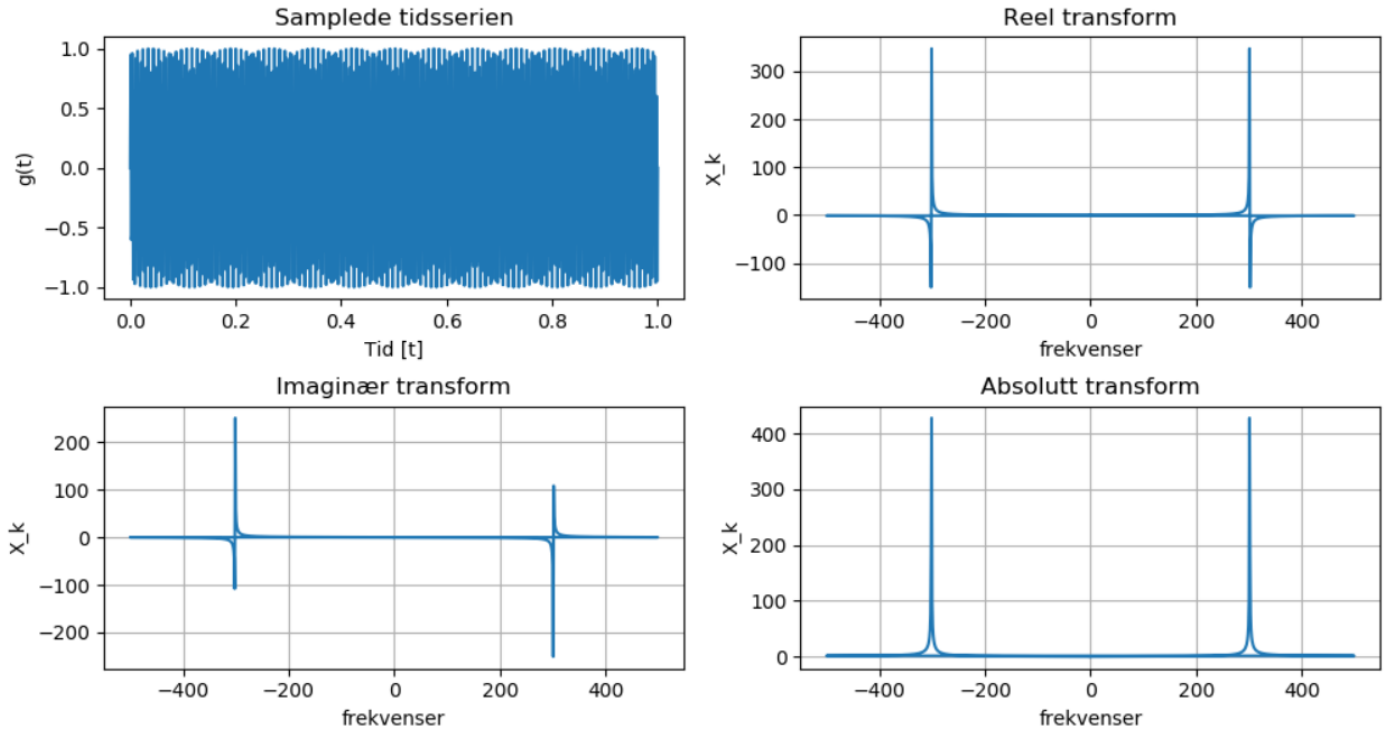


Figure 3: Plot med signalfrekvens på 1.3kHz

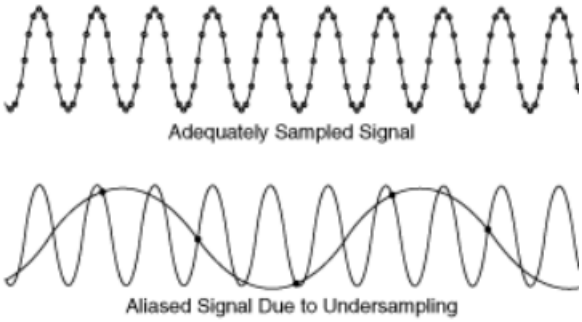
Her kan vi se at ved fourier transform kan vi gjennkjenne bare 400Hz, mens 700Hz og 1300Hz gis det helt feil.
Forklaring til dette er Nyquist-frekvens teoremet som sier at samplingsfrekvensen skal være minst dobbelt så stor som signalfrekvens.

$$f_{\text{signal}} < 2f_{\text{sampling}}$$

Siden vår samplingsfrekvens er bare 1kHz, vil det si at vi kan studere signaler bare opp til 500Hz! Derfor kunne vi gjenkjenne 400Hz signal, og ikke 700Hz.

Når signalfrekvens er over 500Hz, møter vi aliaserings problemet. Aliasing oppstår når vi skal definere et signal med punkter, hvis vi setter 2 punker per 1 periode i signalet, vil det gå fint (kvaliteten vil være ganske liten da, jo flere punkter på en periode desto bedre vil kvaliteten være). Hvis vi har bare 1 punkt i løpet av f.eks 2 perioder, vil den nye opprettet digital signal være helt forskjellig enn den opprinnelig.

Figur nedefor viser akkurat dette.



Oppgave 2

I denne oppgaven skal studere kort Fourier serier. Fourier serier sier at man kan konstruere hvilket som helst periodisk funksjon $f(x)$ ved å summere uendelig mange periodiske funksjoner (sinus og cosinus funksjoner),

$$f(t) = a_0 + a_1\cos(\omega_1 t) + b_1\sin(\omega_1 t) + a_2\cos(\omega_2 t) + b_2\sin(\omega_2 t) + ... + a_n\cos(\omega_n t) + b_n\cos(\omega_n t)$$

I denne oppgaven skal vi vise matematisk at første komponent (altså a_0) av et signal er lik gjennomsnittetsverdien til signalet vi starter med.

Vi kan uttrykke gjennomsnittsverdi til en funksjon ved å skrive,

$$\frac{1}{T}\int_0^T f(t)dt = \bar{f}$$

Hvor T er perioden. Ved denne integralet finner vi gjennomsnitsverdi i løpet av denne integralen.

Vi har sagt at vi kan uttrykke dette periodiske funksjonen med fourier serie. Dermed kan vi erstette $f(x)$ med fourier serie,

$$\frac{1}{T}\int_0^T a_0 + a_1\cos(\omega_1 t) + b_1\sin(\omega_1 t) + a_2\cos(\omega_2 t) + b_2\sin(\omega_2 t) + ... + a_n\cos(\omega_n t) + b_n\cos(\omega_n t)dt$$

Her ser vi at vi har mange ledd, vi kan derfor dele opp integralen til uendelig mange integralet som tilhører til sitt ledd av fourier serien.

Vi kan merke oss en viktig ting, nemlig at alle ledd unntatt a_0 nuller seg ut! Siden hvis man har en periode for en sinus (samme gjelder for sinus), så får man

negative og positive verdier, ved å se det grafisk, kan man se at like mye areal er over x-aksen som under x-aksen. Dermed total arealer blir lik 0!

Vi ender opp med,

$$\frac{1}{T}\int_0^T f(t)dt = \bar{f} = \frac{1}{T}\int_0^T a_0 dt = \frac{1}{T}a_0(T) = a_0$$

Dermed ser vi at $\bar{f} = a_0$!

Til slutt, skal vi vise dette numerisk. I dette tilfelle skal vi velge vår $f(t)$ til å være $\sin^2(t)$. For å vise dette, skal vi bruke fft-funksjonen i python til å finne den første komponenten, til slutt skal vi sammenligne denne funnet komponenten med gjennomsnittsverdi til $\sin^2(t)$. Scriptet er lagt som vedlegg under "Script til Oppgave 2".

Oppgave 3: Waveletsanalyse

I denne oppgaven skal vi lage to bølgepakker i tid, med to forskjellige frekvenskomponenter.

Vi bruker gausiske omhyllingsfunksjoner, og funksjonen som vi skal implementere er $f(t)$ som er gitt i oppgaveteksten.

Vi starter med å plotte den samplede tidsserien for $f(t)$ (med samme samplingsfrekvens og tid), og finne dens diskrete fouriertransform.

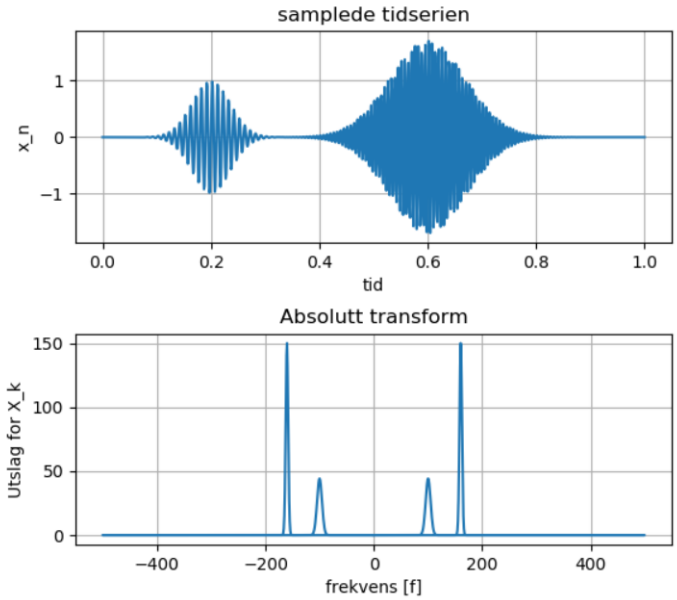


Figure 4: Plot av sampletede tidserien (øverst) og dens diskrete fouriertransform (nederst)

Se vedlegg (Script til Oppgave 3) nederst for å se koden til script for plotting og beregning. I vedlegget står det skrevet hvordan vi wavelettransformerer denne lydfilen slik at kan se hvilket frekvenser har hendt i hvilket tid.

Ved å bruke script til wavelettransformerte av signalet ender vi opp med 2d colorplot som viser tiden i x-aksen, og frekvens i y-aksen.

Vi skal prøve å variere K verdien i vårt transformering. K verdi bestemmer hvor "bred" vår wavelet er som scanner over signalet. Ved stor K ender vi opp med presis representasjon for frekvens, men liten presisjon for tiden denne

frekvensen intreffer. For liten K verdi derimot, er det omvendt.

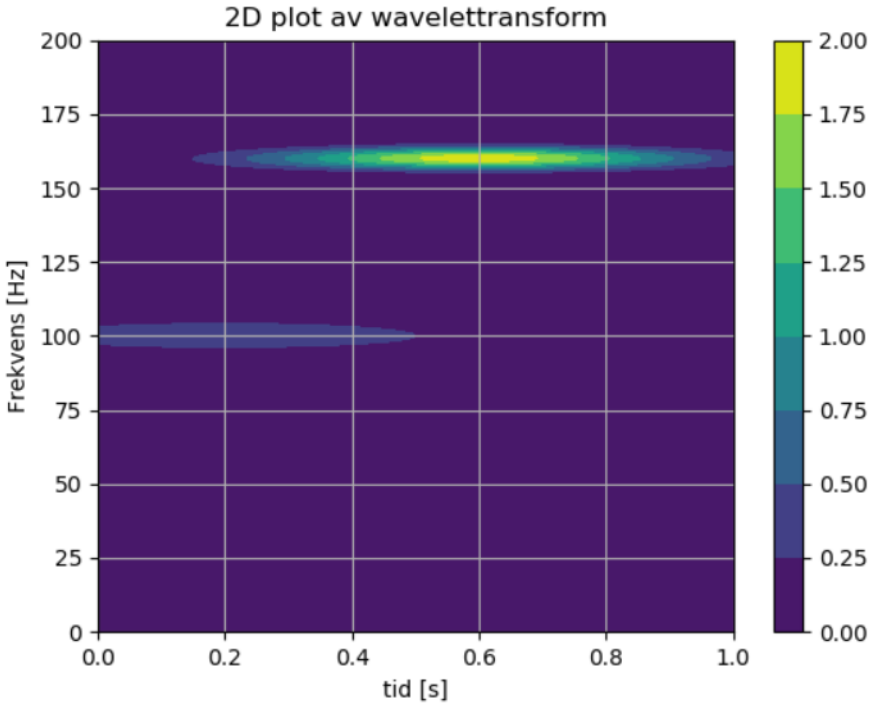


Figure 5: Wavelettransform av signalet med K = 60

På figur 5 kan vi se plot med stor K. Vi kan se utifra figuren at det er ganske vanskelig å si konkret i hvilke tidspunkter visse frekvenser intreffer. Dette skilles pga stor K verdi. Allikevel, vi kan godt si hvilke frekvenser det er, nemlig $f_1 = 100Hz$ og $f_2 = 160$.

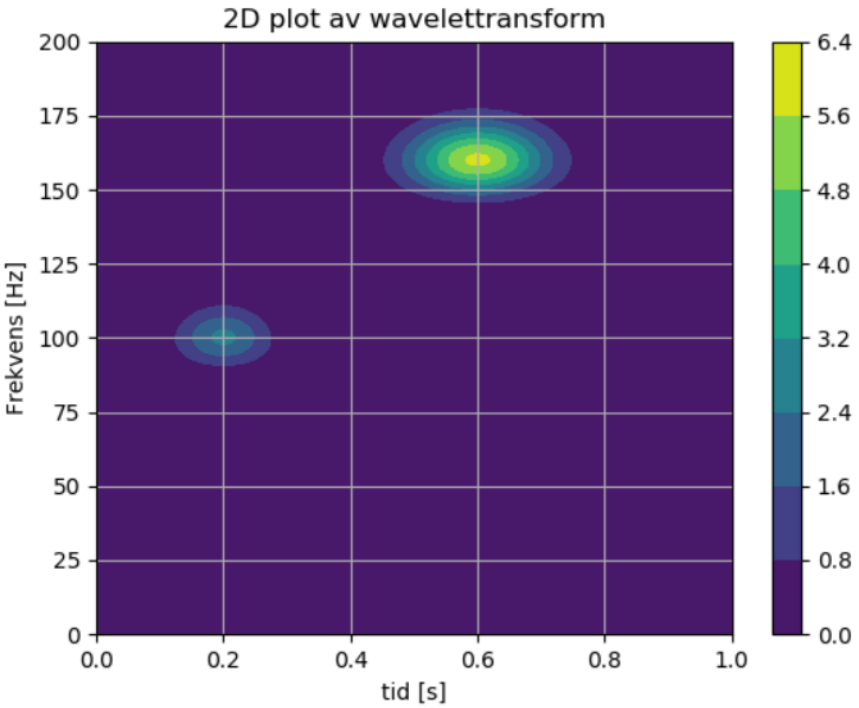


Figure 6: Wavelettransform av signalet med K = 6

I figur 6 kan vi se at utslagene i plottet er ikke like breie som i figur 5. Dette gjør at vi kan enklere lese av tidspunkter. Vi kan nå si at 100Hz intreffet etter 0.2 sekunder, og 160Hz intreffer etter 0.6 sekunder.

Oppgave 4: Waveletanalyse av gjøkegal

Hensikten med denne oppgaven er ganske lik den forrige oppgave. I denne oppgaven skal vi også studere et signal, som vi skal finne wavelettransform til. Eneste forskjellen her er at vi skal bruke et lydfil, som vi skal implementere inn i vårt python script som sampler det.

Vi starter med å skrive en script (ligger som vedlegg "Script til oppgave 4") som sampler lydfil og utfører fouriertransformasjon.
I figur 7 kan vi se fouriertransform av signalet, utifra den kan vi si at vi har

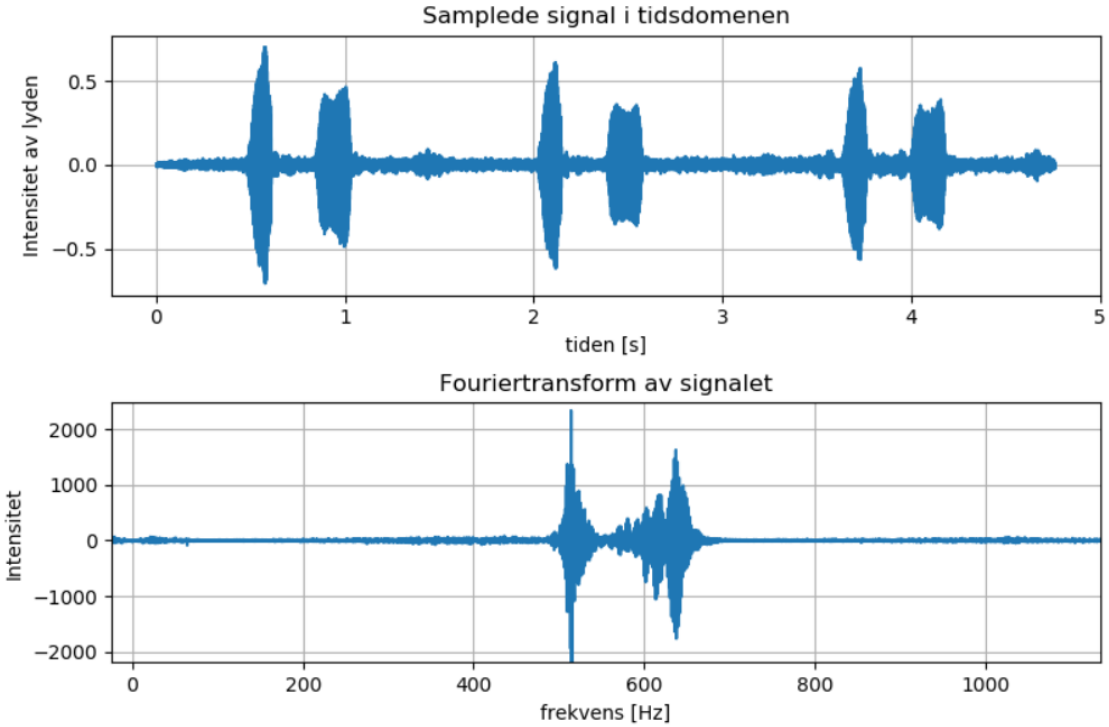


Figure 7: Samplede signalet (Øverst) og fouriertransform av signalet (Nederst)

to frekvenser som virker inn i lydfilen. Den første ligger rundt 500Hz, og den andre rundt 620Hz (Merk at dette er bare visuell estimering). Vi kan itillegg merke at "coo-coo" lyden skrev periodisk. Vi kan derfor velge et intervall av bare den ene "coo-coo", f.eks mellom tid 0.4-1.1 sek.

Vi skriver script som finner ut samplingfrekvensen. Vi finner ut at denne samplingsfrekvensen er på 44.1kHz, som er standard for lyd av CD-kvalitet. Mennesker kan høre opp til 20kHz, all lyd som har høyere frekvens enn der, er det umulig for oss å høre. Dermed Nyquist-frekvensen for det er på 40kHz. Det derfor standard CD-samplingsfrekvens er høyrere enn det. Det spiller ingen rolle for oss om vi får aliasing over 20kHz, siden vi ikke kan høre det.

Vi skriver en script som waveletanalyserer denne signalet. Som resultat ender vi opp med 2D colorplot som viser i hvilket tidspunker hvilke frekvens intreffer. Vi starter med å sette K-verdi lik 100 for å finne mer presist hvilke frekvenser signalet intreffer.

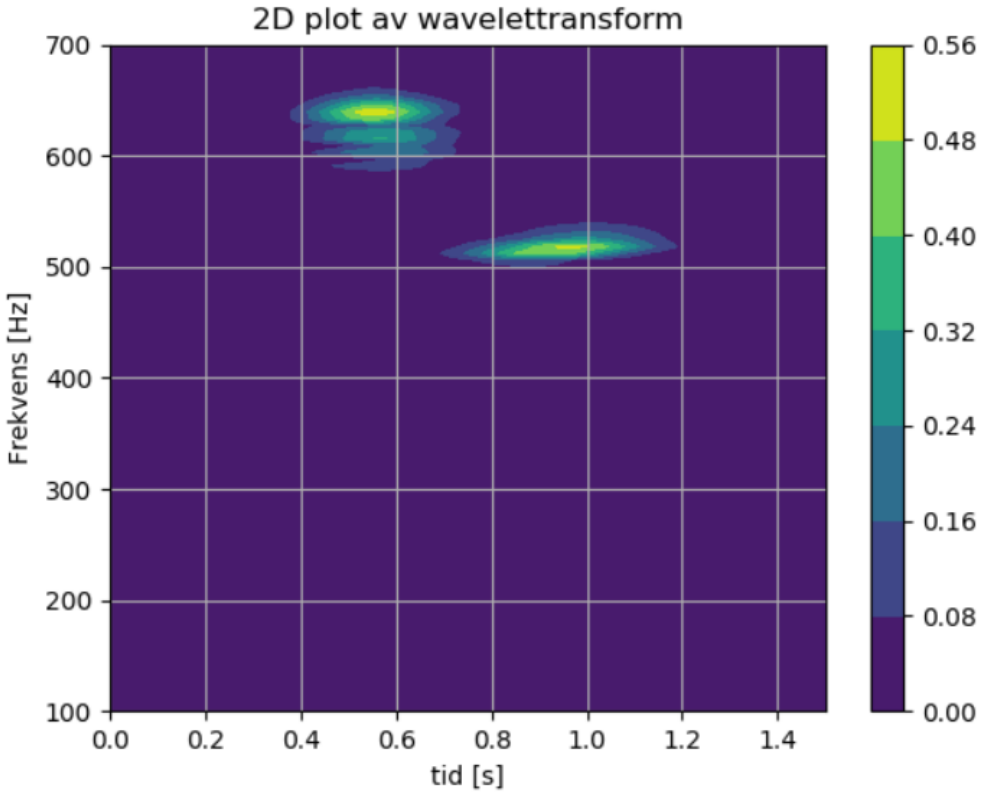


Figure 8: Wavelettransform av signalet med K = 100

I figure 8 kan vi observere at vi har 2 frekvenser som intreffer. Dette er akkurat det vi forventer, siden vi har sett det i fouriertransform. Utifra plottet kan vi lese av at den første frekvensen er på rundt 640Hz, og den andre er på rundt 518Hz.

Vi velger nå en mindre K verdi, slik at vi skal kunne lese mer presis tidspunkter når disse frekvensene intreffer. Vi velger K til å være 5.

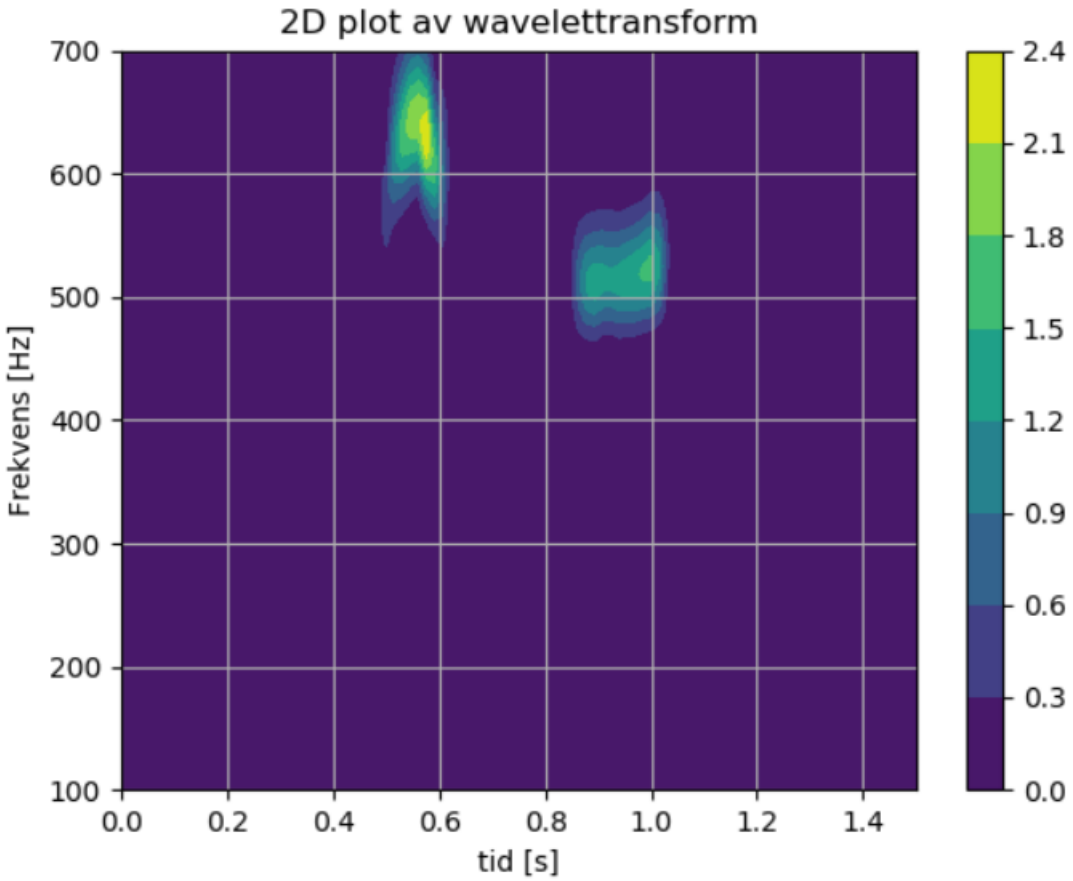


Figure 9: Wavelettransform av signalet med K = 5

Utifra figur 9 kan vi se at signalet er på 640Hz etter rundt 0.57s, og 518Hz etter rundt 0.99s.

SCRIPT til Oppgave1

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

def plot(x,y,xlabel,ylabel,title):
    """
    Lager plottingsfunksjon
    variablene xlabel, ylabel og title skal ha skriver med ""
    """
    plt.plot(x,y)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title()
    plt.grid(True)


#Signal parametere
f_sig = --- #Signalfrekvens [Hz] #!!!HER MAA MAN SETTE INN VERDI !!!#
A = 1 #Amplituden [m]

#Sampling parametere
T = 1 #Samplingstiden [s]
f_samp = 1000 #Samplingsfrekvens [Hz]
N = T*f_samp #Antall samplings punkter i l pet av 1 sekund
dt = T/N #Tiden mellom samplingspunkter

t = np.linspace(0, T, N) #Definerer en array der jeg lagrer alle tidsverdier
x_n = A*np.sin(2*np.pi*f_sig*t) #Signalverdier for hver samplings delta tider

plt.subplot(221)
plot(t,x_n,'Tid [t]','g(t)','Samplede tidsserien')

X_k = np.fft.fft(x_n) #fft kommando for fourier transforme x_n punktene
freq = np.fft.fftfreq(N,dt) #fftfreq kommando for    unngaa folding

plt.subplot(222)
plot(freq,np.real(X_k),'frekvenser','X_k','Reel transform of signal with ---Hz')
plt.subplot(223)
plot(freq,np.imag(X_k),'frekvenser','X_k','imaginary transform of signal with ---Hz')
plt.subplot(224)
plot(freq,abs(X_k),'frekvenser','X_k','absolute transform of signal with ---Hz')
plt.show()

###FOR OPPGAVE1B SAA DEFINERER MAN NYE VERDIER FOR F_S OG KJOORER SCRIPT PAA NYTT###
```

SCRIPT til Opggave 2

```
import numpy as np
import matplotlib.pyplot as plt
N = 1000 #Velger tilfeldig N antall

def f(t):
    return np.sin(t)**2


t = np.linspace(0,np.pi,N) #Velger en periode for sinus, som er pi


def avarage(): #Finner gjennmsnitverdi
    sum = 0
    for i in range(N):
        sum = sum + f(t[i])
    return (1/np.pi)*sum

DFT = np.real((1/np.pi)*np.fft.fft(f(t)))

print(f(f"Manuell regnet gjennomsnitt:{avarage()}"))
print(f"F rste komponent funnet ved fft kommando: {DFT[0]}")
```

SCRIPT til oppgave 3

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

A_1 = 1.0 #[m]
A_2 = 1.7 #[m]
f_1 = 100 #[Hz]
f_2 = 160 #[Hz]
t_1 = 0.20 #[s]
t_2 = 0.60 #[s]
sigma_1 = 0.05 #[s]
sigma_2 = 0.10 #[s]
f_s = 1000 #Samplingsfrekvens [Hz]
T = 1 #Samplingstid [s]
N = T*f_s #Antall samplings punkter i loopet av 1 sekund
dt = T/N #Tiden mellom samplingspunkter


def f(t): #Definerer funksjonen som vi skal studere og som er oppgitt i oppgaveteksten
    ledd1 = A_1*np.sin(2*np.pi*f_1*t)*np.exp(-((t - t_1)/sigma_1)**2)
    ledd2 = A_2*np.sin(2*np.pi*f_2*t)*np.exp(-((t - t_2)/sigma_2)**2)
    return ledd1 + ledd2

t = np.linspace(0,T,N) #Definerer en array der jeg lagrer alle tidsverdier
n_x = np.zeros(N) #Definerer tom array som vi skal fylle inn med signalverdien for hver delta samplingstid
for i in range(N): #Bruker for-loop til    fylle n_x
    n_x[i] = (f(t[i]))


def wavelet(f_a, K, t_k, t_n): #Definerer wavelet funksjon som er oppgitt i ligning 14.8 i kompendiet
    C = 0.798*2*np.pi*f_a/(f_s*K) #Fant C konstanten i kompendium
    eksp1 = -1j*2*np.pi*f_a*(t_n - t_k)
    eksp2 = -K**2
    eksp3 = (-2*np.pi*f_a**2)*((t_n - t_k)**2)/(2*K)**2
    return C*(np.exp(eksp1) - np.exp(eksp2))*np.exp(eksp3)

def gamma(f_a, K, t_k): #Definerer denne fra ligningen 14.9 som er i kompendiet
    sum = 0
    for i in range(N):
        sum = sum + np.conj(n_x[i]*(wavelet(f_a, K, t_k, t[i])))
    return sum

K = --- #HER SKAL DU VELGE K VERDI  F  R  MAN  KJ RER  SCRIPT!#

#2D colorplotting:

t_ = np.linspace(0,1,700) #Velger array med alle tidsverdier vi vil teste ut
```

```
f_ = np.linspace(0,200,100) #Velger array med alle frekvenser vi
    vil teste ut
T,F = np.meshgrid(t_,f_) #Gj r disse om til en matrise
Z = abs(gamma(F,K,T)) #Regner ut utslag til gamma for hver
    komponent i matrisen
plt.contourf(T,F,Z) #Plotter denne matrisen i 2d som colorplot
#Navnsetting av akser og pengj ring :)
plt.grid(True)
plt.colorbar()
plt.xlabel("tid [s]")
plt.ylabel("Frekvens [Hz]")
plt.title("2D plot av wavelettransform")
plt.show()
```

SCRIPT til Oppgave 4a

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile

samplerate, data = wavfile.read('cuckoo.wav')
x_n = data[:,0] #Siden dette er en stereo fil, velger man en av de
    kanalene
N = data.shape[0] #finner ut antall samples
f_samp = samplerate #samplefrekvens [Hz]
T = N/f_samp #Sampletiden
dt = 1/f_samp

print(f"Samplingsfrekvensen er p   {f_samp}Hz")

t = np.linspace(0,T,N)

X_k = np.fft.fft(x_n)
freq = np.fft.fftfreq(N,dt)
plt.subplot(211)
plt.plot(t,x_n)
plt.xlabel("tiden [s]")
plt.ylabel("Intensitet av lyden")
plt.title("Samplede signal i tidsdomenen")
plt.grid(True)
plt.subplot(212)
plt.plot(freq, X_k)
plt.xlabel("frekvens [Hz]")
plt.ylabel("Intensitet")
plt.title("Fouriertransform av signalet")
plt.grid(True)
plt.show()
```

SCRIPT til Oppgave 4b

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile


samplerate, data = wavfile.read('cuckoo.wav')
x_n_full = data[:,0] #Siden dette er en stereo fil, velger man en
    av de kanalene
N_full = data.shape[0] #finner ut antall samples
f_s = samplerate #samplefrekvens [Hz]
T_full = N_full/f_s #Sampletiden
dt = 1/f_s

t_full = np.linspace(0,T_full,N_full)


t = t_full[18000:50000] #Slicer opp t slik at vi kan fokuserer bare
     p   det ene coo-coo
x_n = x_n_full[18000:50000] #Slicer den ogs .
N = t.shape[0]
T = N/f_s

def wavelet(f_a, K, t_k, t_n):
    C = 0.798*2*np.pi*f_a/(f_s*K)
    eksp1 = -1j*2*np.pi*f_a*(t_n - t_k)
    eksp2 = -K**2
    eksp3 = (-2*np.pi*f_a**2)*((t_n - t_k)**2)/(2*K)**2
    return C*(np.exp(eksp1) - np.exp(eksp2))*np.exp(eksp3)

def gamma(f_a, K, t_k):
    sum = 0
    for i in range(N):
        sum = sum + np.conj(x_n[i]*(wavelet(f_a, K, t_k, t[i])))
    return sum


K = 5

t_ = np.linspace(0,1.5,100)
f_ = np.linspace(100,700,100)
T,F = np.meshgrid(t_,f_)
Z = abs(gamma(F,K,T))
plt.contourf(T,F,Z)
plt.grid(True)
plt.colorbar()
plt.xlabel("tid [s]")
plt.ylabel("Frekvens [Hz]")
plt.title("2D plot av wavelettransform")
plt.show()
```