Obligatorisk innlevering i IN3190/IN4190

Signalbehandling i Seismikk

klaudiap

07. oktober 2020

Oppgave 1 (Konvolusjon og frekvensspekter)

a. Vi bruker Python og skriver følgende kode:

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

#Oppgave 1A
def konvin3190(h,x,ylen):

    """
    konvin3190 konvolverer to signaler
    konvin3190(h,x,ylen) konvolverer de to vektor-signalene h og x.
    Dersom ylen = 0 er utgangssignalet lengde "len(x)", mens hvis ylen
    er 1 har utgangssignalet lengde "len(h)+len(x)-1"
    """

    x_len = len(x)
    h_len = len(h)

    if ylen == 1:
        y = np.zeros(x_len+h_len-1)
        for i in range(h_len):
            for j in range(x_len):
                k = j+i-1
                y[k] = y[k]+h[i]*x[j]

    if ylen == 0:
        y = np.zeros(x_len)
        for i in range(h_len):
            for j in range(x_len-h_len+1):
                k=i+j-1
                y[k] = y[k]+h[i]*x[j]
    return y
```

b. Vi bruker Python og skriver følgende kode (vi fortsetter på tidligere programmet):

```
#Oppgave 1B
def frekspekin3190(x,N,fs):

"""
```

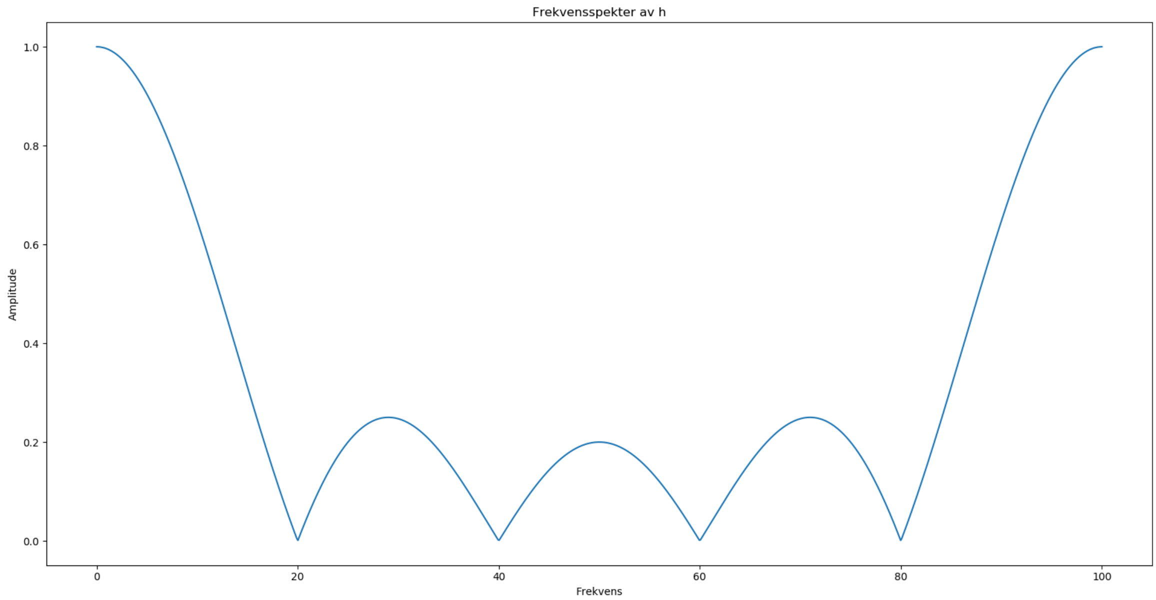
```
frekspekin3190 regner ut frekvensresponsen til et signal
x,g = frekspekin3190(x,N,fs) regner ut frekvensresponsen til
signalet x med samplingfrekvens fs for N punkter på enhetssirkelen
I tillegg til frekvensspekteret X returnerer funksjonen også tilhørende
frekvens f
"""

x_len = len(x)
X = np.zeros((N),dtype = "complex_")
fp = np.linspace(0,1,N)
f = fp*fs

for i in range(N):
   for j in range(x_len):
      X[i] = X[i]+x[j]*np.exp(2*np.pi*-1j*fp[i]*j)

return X,f
```

c. Vi bruker Python og skriver følgende kode (vi fortsetter på tidligere programmet):

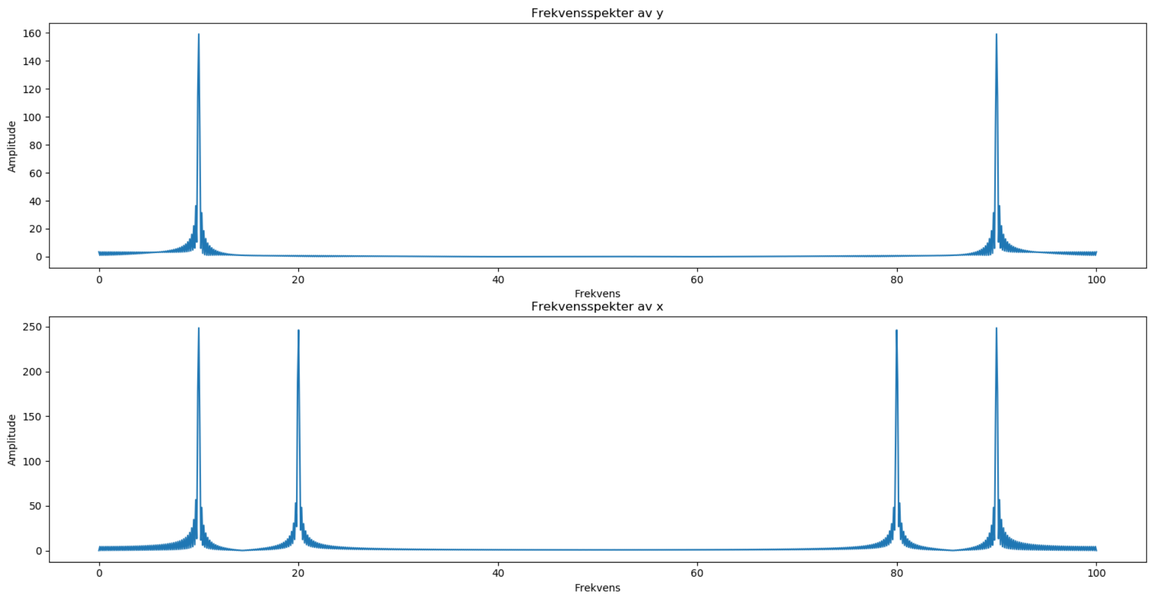
```
#Oppgave 1C
f1 = 10
f2 = 20
fs = 100
t_len = 5

t = np.linspace(0,t_len,t_len*fs+1)
h = [0.2,0.2,0.2,0.2,0.2] #FIR filter
N = 1000 #Vi velger antall punkter
x = np.sin(2*np.pi*f1*t)+np.sin(2*np.pi*f2*t) #Den gitte signalet

Xh,fh = frekspekin3190(h,N,fs) #Frekvensspekter av h
Xx,fx = frekspekin3190(x,N,fs) #Frekvensspekter av x
Xy,fy = frekspekin3190(konvin3190(h,x,0),N,fs) #Frekvensspekter av y

plt.plot(fh,abs(Xh))
plt.title('Frekvensspekter av h')
plt.xlabel('Frekvens')
plt.ylabel('Amplitude')
plt.show()

plt.subplot(211);
plt.plot(fy,abs(Xy))
plt.title('Frekvensspekter av y')
plt.xlabel('Frekvens')
plt.ylabel('Amplitude')

plt.subplot(212);
plt.plot(fx,abs(Xx))
```

```python
plt.title('Frekvensspekter av x')
plt.xlabel('Frekvens')
plt.ylabel('Amplitude')
plt.show()
```

Og vi får:



Og:



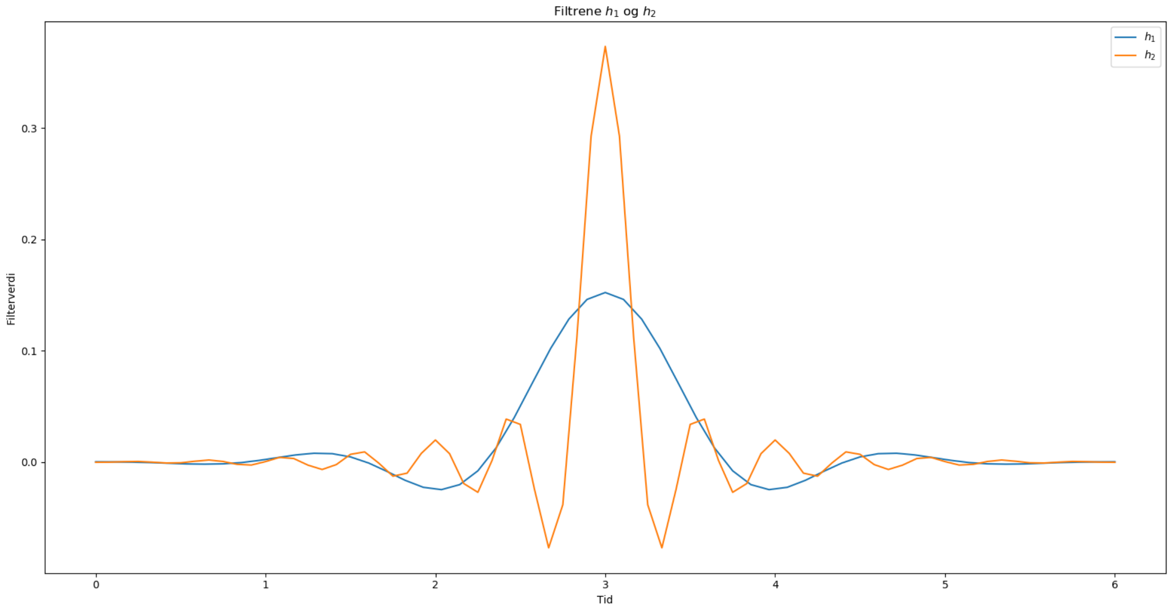
Vi får en topp ved $10Hz$ på frekvensspekter av y, og to topper $10Hz$ og $20Hz$ på frekvensspekter av x (de andre toppene er en speiling). Vi ser at toppen på $20Hz$ ble filtrert ut (pga. konvolusjon), samtidig som amplituden har blitt mindre. Dette stemmer med frekvensspekter av h, da amplituden er 0 ved $20Hz$.

Oppgave 2 (Støyfjerning)

a. Vi bruker Python og plotter h_1 og h_2 i samme figur:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio

#Oppgave2A
klaudiap = sio.loadmat('klaudiap.mat')

t = klaudiap['t']
seismogram1 = klaudiap['seismogram1']
seismogram2 = klaudiap['seismogram2']
offset1 = klaudiap['offset1']
offset2 = klaudiap['offset2']

h1 = [0.0002, 0.0001, -0.0001, -0.0005, -0.0011, -0.0017, -0.0019,
      -0.0016, -0.0005, 0.0015, 0.0040, 0.0064, 0.0079, 0.0075, 0.0046,
      -0.0009, -0.0084, -0.0164, -0.0227, -0.0248, -0.0203, -0.0079,
      0.0127, 0.0400, 0.0712, 0.1021, 0.1284, 0.1461, 0.1523, 0.1461,
      0.1284, 0.1021, 0.0712, 0.0400, 0.0127, -0.0079, -0.0203, -0.0248,
      -0.0227, -0.0164, -0.0084, -0.0009, 0.0046, 0.0075, 0.0079, 0.0064,
      0.0040, 0.0015, -0.0005, -0.0016, -0.0019, -0.0017, -0.0011,
      -0.0005, -0.0001, 0.0001, 0.0002]

h2 = [-0.0002, -0.0001, 0.0003, 0.0005, -0.0001, -0.0009, -0.0007,
      0.0007, 0.0018, 0.0005, -0.0021, -0.0027, 0.0004, 0.0042, 0.0031,
      -0.0028, -0.0067, -0.0023, 0.0069, 0.0091, -0.0010, -0.0127,
      -0.0100, 0.0077, 0.0198, 0.0075, -0.0193, -0.0272, 0.0014, 0.0386,
      0.0338, -0.0246, -0.0771, -0.0384, 0.1128, 0.2929, 0.3734, 0.2929,
      0.1128, -0.0384, -0.0771, -0.0246, 0.0338, 0.0386, 0.0014, -0.0272,
      -0.0193, 0.0075, 0.0198, 0.0077, -0.0100, -0.0127, -0.0010, 0.0091,
      0.0069, -0.0023, -0.0067, -0.0028, 0.0031, 0.0042, 0.0004, -0.0027,
      -0.0021, 0.0005, 0.0018, 0.0007, -0.0007, -0.0009, -0.0001, 0.0005,
      0.0003, -0.0001, -0.0002]

t1 = np.linspace(np.amin(t),np.amax(t),len(h1))
t2 = np.linspace(np.amin(t),np.amax(t),len(h2))

plt.plot(t1,h1)
plt.plot(t2,h2)
plt.title('Filtrene $h_1$ og $h_2$')
plt.xlabel('Tid')
plt.ylabel('Filterverdi')
plt.legend(['$h_1$','$h_2$'])
plt.show()
```

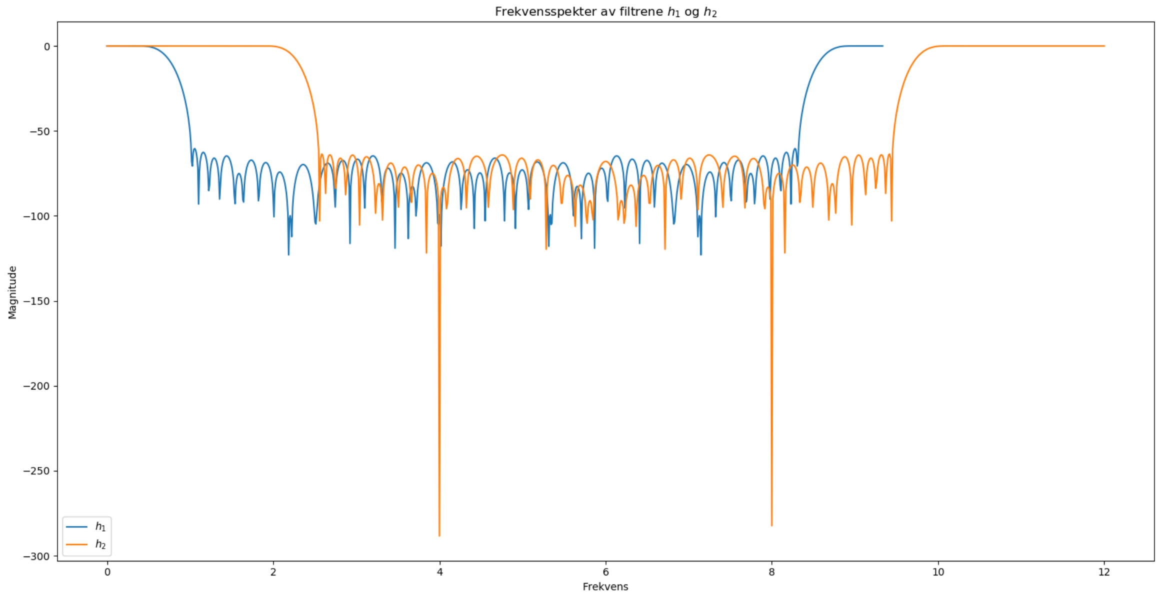
Og får:



Vi skriver videre på programmet for å regne ut frekvensspekteret til h_1 og h_2:

```
def frekspekin3190(x,N,fs):
    x_len = len(x)
    X = np.zeros((N),dtype = "complex_")
    fp = np.linspace(0,1,N)
    f = fp*fs

    for i in range(N):
        for j in range(x_len):
            X[i] = X[i]+x[j]*np.exp(2*np.pi*-1j*fp[i]*j)

    return X,f

N = 1000
fs1 = 1/(t1[2]-t1[1])
fs2 = 1/(t2[2]-t2[1])

H1,fh1 = frekspekin3190(h1,N,fs1)
H2,fh2 = frekspekin3190(h2,N,fs2)

plt.plot(fh1,20*np.log10(abs(H1)),fh2,20*np.log10(abs(H2)))
plt.title('Frekvensspekter av filtrene $h_1$ og $h_2$')
plt.xlabel('Frekvens')
plt.ylabel('Magnitude')
plt.legend(["$h_1$","$h_2$"])
plt.show()
```

Og får

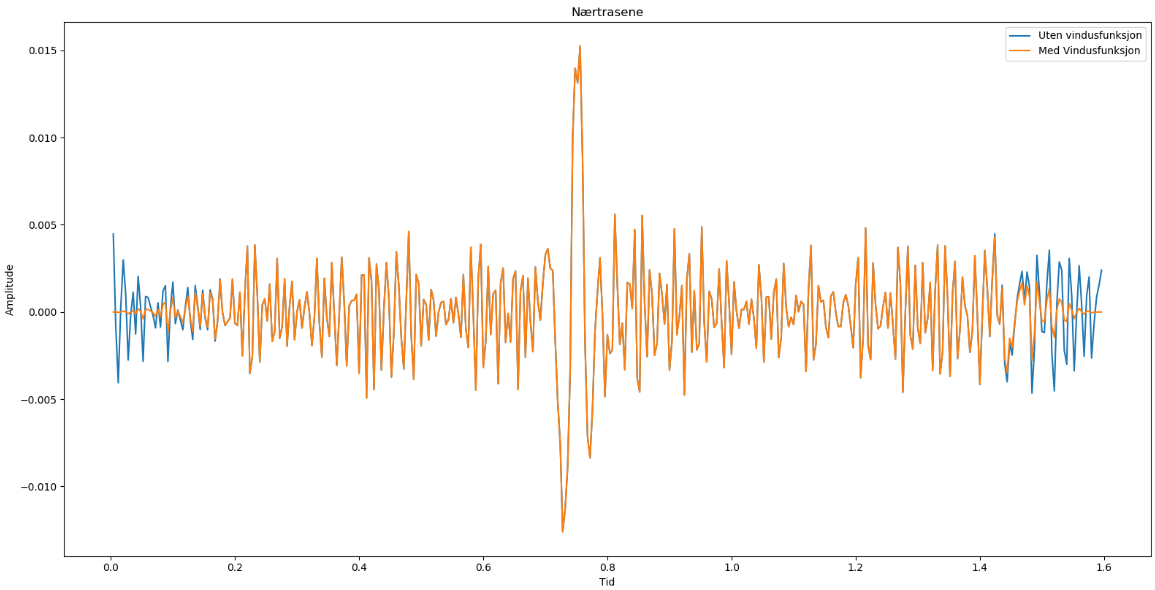


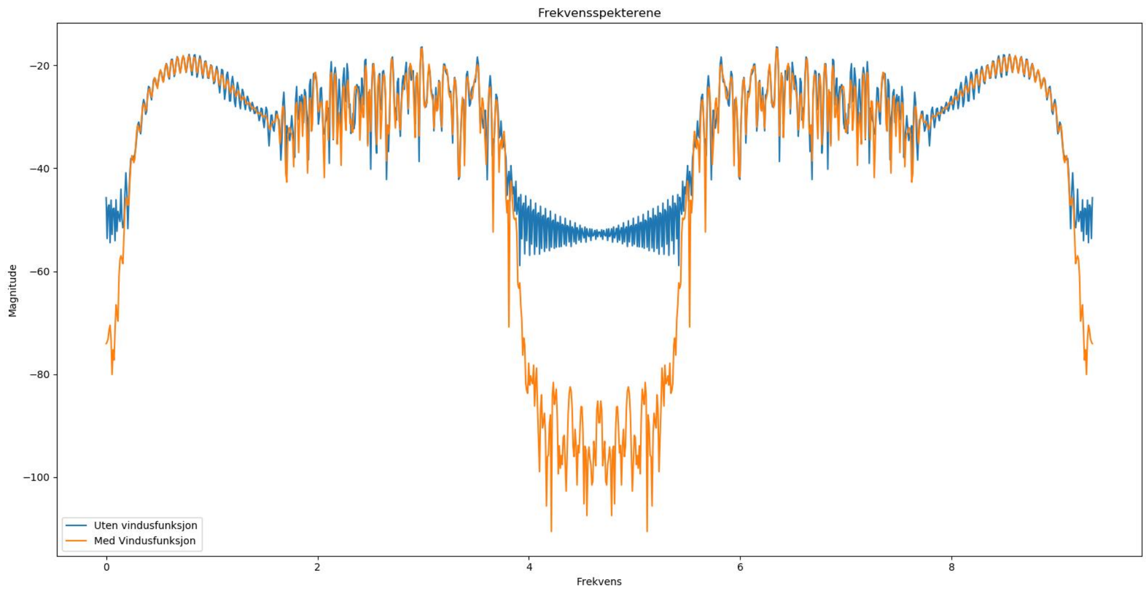
Vi ser filter h_1 kan filtrere mer frekvenser enn h_2. h_1 filtrerer de lave frekvenser (fra $1Hz$ til $2.5Hz$), men ikke de høye frekvenser ($8.2Hz$ til $9.5Hz$), og h_2 filtrerer de høye frekvenser ($8.2Hz$ til $9.5Hz$), men ikke de lave frekvenser (fra $1Hz$ til $2.5Hz$).

b. Vi skriver videre på forrige programmet, og får:

```
from scipy import signal

s = seismogram1[1:400,1]
w = signal.tukey(len(s),0.25)

sw = s*w

plt.plot(t[1:400], s)
plt.plot(t[1:400], sw)
plt.title('Nærtrasene')
plt.xlabel('Tid')
plt.ylabel('Amplitude')
plt.legend(["Uten vindusfunksjon","Med Vindusfunksjon"])
plt.show()

fs,f = frekspekin3190(s,N,fs1)
fsw,fw = frekspekin3190(sw,N,fs1)

plt.plot(f,20*np.log10(abs(fs)))
plt.plot(fw,20*np.log10(abs(fsw)))
plt.title('Frekvensspekterene')
plt.xlabel('Frekvens')
plt.ylabel('Magnitude')
plt.legend(["Uten vindusfunksjon","Med Vindusfunksjon"])
plt.show()
```

Kjøreeksempel gir:





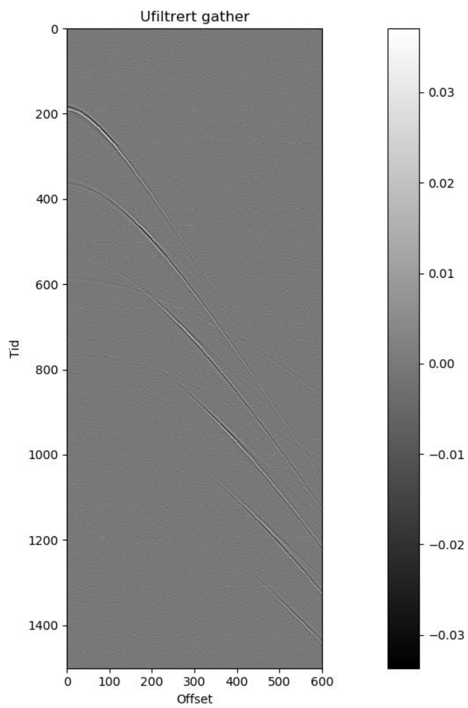
De «svingigene» vi ser på plottet (gjennom hele lengden av plottet) representerer støy, resten representerer signal.

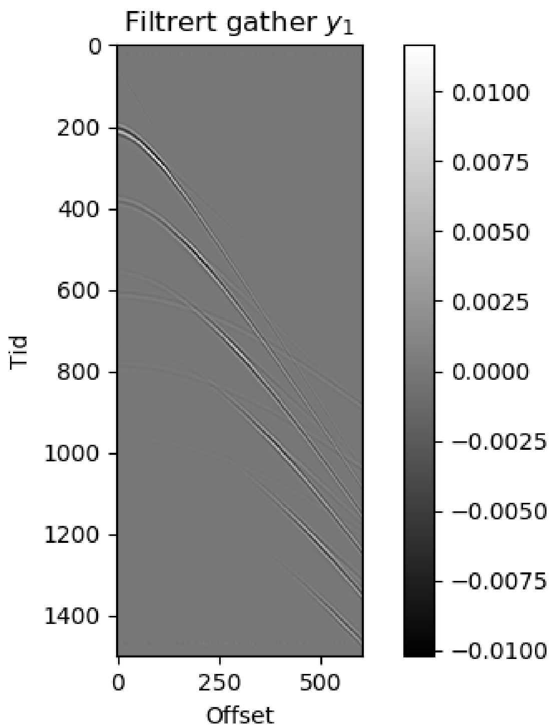
c. Vi bruker funksjonen konvin3190 fra oppgave 1, og skriver videre på forrige programmet. Vi får:

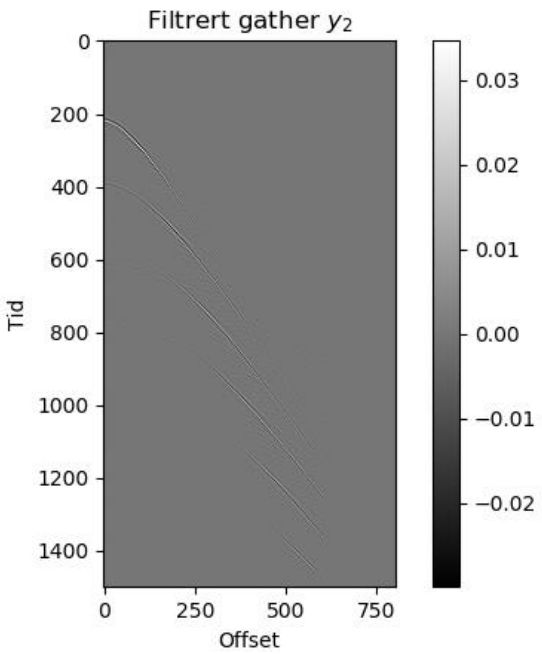
```
def konvin3190(h,x,ylen):
    x_len = len(x)
    h_len = len(h)

    if ylen == 1:
        y = np.zeros(x_len+y_len-1)
        for i in range(h_len):
            for j in range(x_len):
```

```
              k = j+i-1
              y[k] = y[k]+h[i]*x[j]

    if ylen == 0:
      y = np.zeros(x_len)
      for i in range(h_len):
          for j in range(x_len-h_len+1):
              k=i+j-1
              y[k] = y[k]+h[i]*x[j]
    return y

y1 = np.zeros_like(seismogram1)
y2 = np.zeros_like(seismogram2)
m,n = np.shape(seismogram1)

for i in range(n):
    y1[:,i] = konvin3190(h1,seismogram1[:,i],0)
    y2[:,i] = konvin3190(h2,seismogram1[:,i],0)

plt.imshow(seismogram1,cmap='gray')
plt.title('Ufiltrert gather')
plt.xlabel('Offset')
plt.ylabel('Tid')
plt.colorbar()
plt.show()

plt.imshow(y1,cmap='gray')
plt.title('Filtrert gather $y_1$')
plt.xlabel('Offset')
plt.ylabel('Tid')
plt.colorbar()
plt.show()

plt.imshow(y2,cmap='gray')
plt.title('Filtrert gather $y_2$')
plt.xlabel('Offset')
plt.ylabel('Tid')
plt.colorbar()
plt.show()
```

Kjøreeksempel gir:


Ufiltrert gather
Tid
Offset
0
200
400
600
800
1000
1200
1400
0
100
200
300
400
500
600
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03

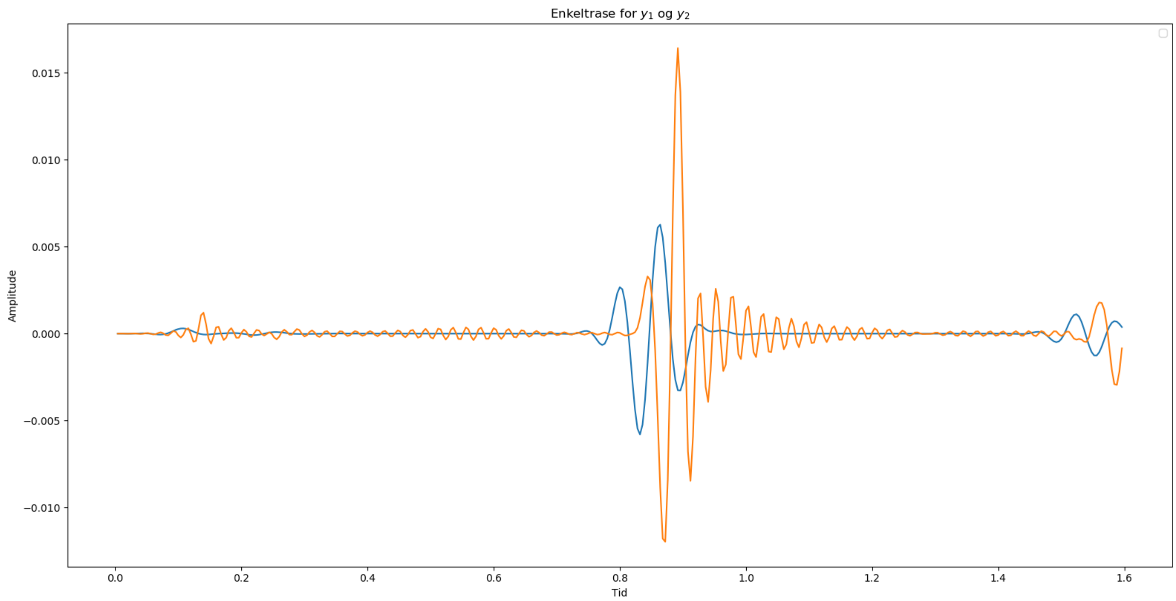

Filtrert gather y_1
Tid
Offset
0
200
400
600
800
1000
1200
1400
0
250
500
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100



Vi fortsetter på programmet:

```
plt.plot(t[1:400], y1[1:400,1],t[1:400],y2[1:400,1])
plt.title('Enkeltrase for $y_1$ og $y_2$')
plt.xlabel('Tid')
plt.ylabel('Amplitude')
plt.legend('$y_1$','$y_2$')
plt.show()
```

Kjøreeksempel gir:



Filter h_1 passer best til støyfjerning for vårt gather. Vi ser ut fra plottet at signalet filtrert ut med filter h_1 (blå) ikke har noe støy.

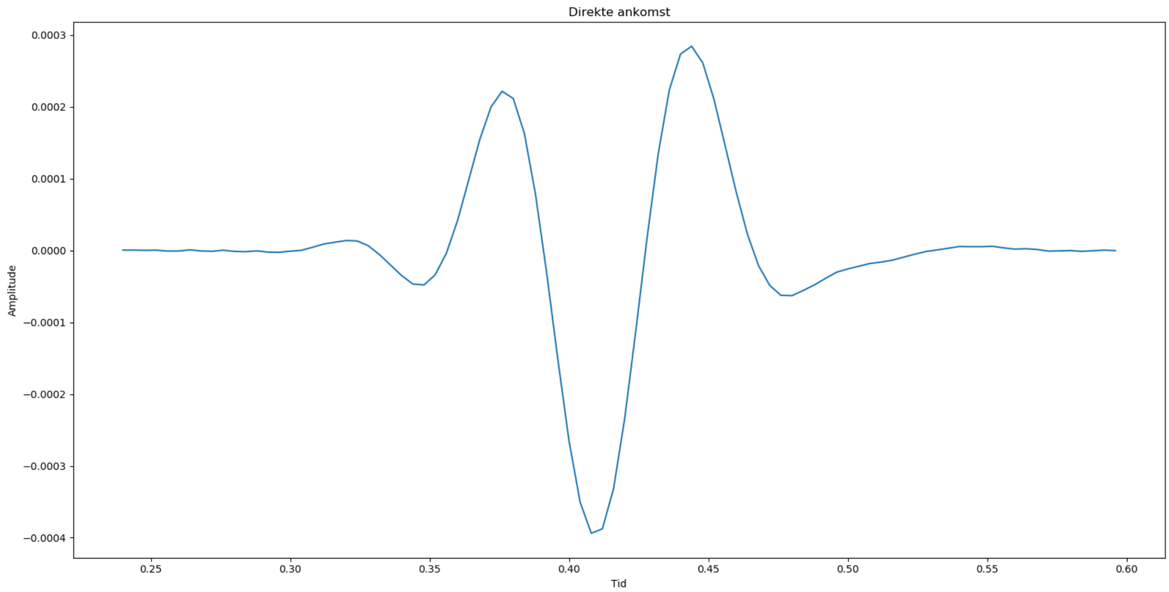
Oppgave 3 (Fjernfeltssignatur)

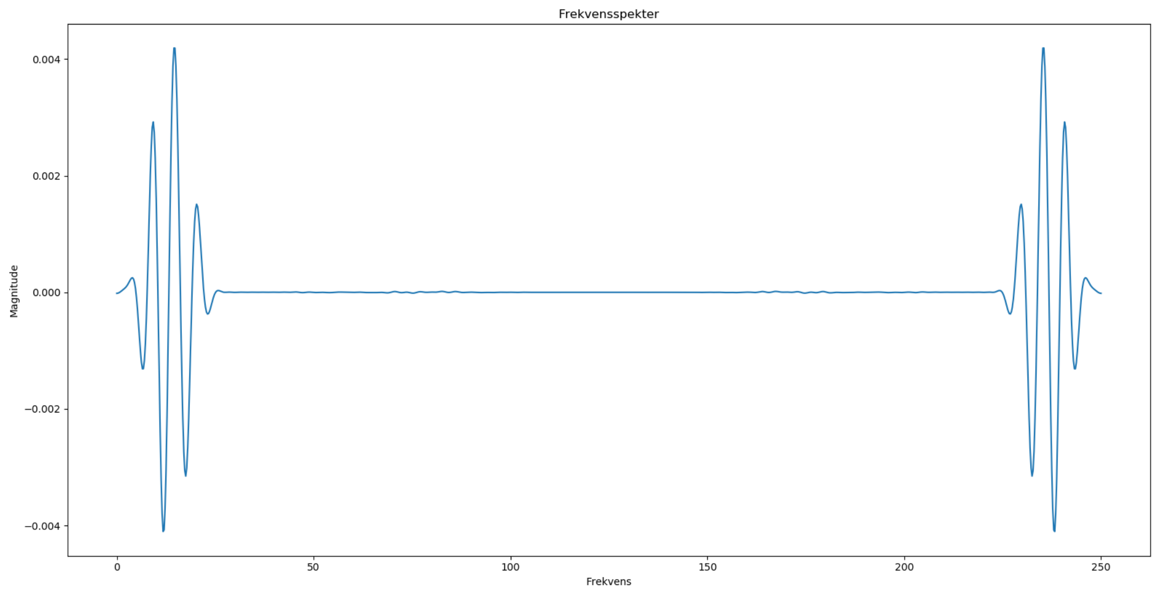
a. Vi skriver programmet i Python, og får:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio

klaudiap = sio.loadmat('klaudiap.mat')

t = klaudiap['t']
seismogram1 = klaudiap['seismogram1']
seismogram2 = klaudiap['seismogram2']
offset1 = klaudiap['offset1']
offset2 = klaudiap['offset2']

h1 = [0.0002, 0.0001, -0.0001, -0.0005, -0.0011, -0.0017, -0.0019,
      -0.0016, -0.0005, 0.0015, 0.0040, 0.0064, 0.0079, 0.0075, 0.0046,
      -0.0009, -0.0084, -0.0164, -0.0227, -0.0248, -0.0203, -0.0079,
      0.0127, 0.0400, 0.0712, 0.1021, 0.1284, 0.1461, 0.1523, 0.1461,
      0.1284, 0.1021, 0.0712, 0.0400, 0.0127, -0.0079, -0.0203, -0.0248,
      -0.0227, -0.0164, -0.0084, -0.0009, 0.0046, 0.0075, 0.0079, 0.0064,
      0.0040, 0.0015, -0.0005, -0.0016, -0.0019, -0.0017, -0.0011,
      -0.0005, -0.0001, 0.0001, 0.0002]

h2 = [-0.0002, -0.0001, 0.0003, 0.0005, -0.0001, -0.0009, -0.0007,
      0.0007, 0.0018, 0.0005, -0.0021, -0.0027, 0.0004, 0.0042, 0.0031,
      -0.0028, -0.0067, -0.0023, 0.0069, 0.0091, -0.0010, -0.0127,
      -0.0100, 0.0077, 0.0198, 0.0075, -0.0193, -0.0272, 0.0014, 0.0386,
      0.0338, -0.0246, -0.0771, -0.0384, 0.1128, 0.2929, 0.3734, 0.2929,
      0.1128, -0.0384, -0.0771, -0.0246, 0.0338, 0.0386, 0.0014, -0.0272,
      -0.0193, 0.0075, 0.0198, 0.0077, -0.0100, -0.0127, -0.0010, 0.0091,
      0.0069, -0.0023, -0.0067, -0.0028, 0.0031, 0.0042, 0.0004, -0.0027,
      -0.0021, 0.0005, 0.0018, 0.0007, -0.0007, -0.0009, -0.0001, 0.0005,
      0.0003, -0.0001, -0.0002]

t1 = np.linspace(np.amin(t),np.amax(t),len(h1))
t2 = np.linspace(np.amin(t),np.amax(t),len(h2))

def konvin3190(h,x,ylen):
    x_len = len(x)
    h_len = len(h)

    if ylen == 1:
        y = np.zeros(x_len+y_len-1)
        for i in range(h_len):
            for j in range(x_len):
                k = j+i-1
                y[k] = y[k]+h[i]*x[j]
```

```

    if ylen == 0:
        y = np.zeros(x_len)
        for i in range(h_len):
            for j in range(x_len-h_len+1):
                k=i+j-1
                y[k] = y[k]+h[i]*x[j]
        return y

def frekspekin3190(x,N,fs):
    x_len = len(x)
    X = np.zeros((N),dtype = "complex_")
    fp = np.linspace(0,1,N)
    f = fp*fs

    for i in range(N):
        for j in range(x_len):
            X[i] = X[i]+x[j]*np.exp(2*np.pi*-1j*fp[i]*j)

    return X,f

N = 1000
fs = 1/(t[2]-t[1])
y1 = np.zeros_like(seismogram1)
m,n = np.shape(seismogram1)

for i in range(n):
    y1[:,i] = konvin3190(h1,seismogram1[:,i],0)
p = y1[60:150,30]

plt.plot(t[60:150],p)
plt.title('Direkte ankomst')
plt.xlabel("Tid")
plt.ylabel("Amplitude")
plt.show()

Sl,sf = frekspekin3190(p, N, fs)

plt.plot(sf, Sl)
plt.title('Frekvensspekter')
plt.xlabel("Frekvens")
plt.ylabel("Magnitude")
plt.show()
```

Kjøreeksempel gir:



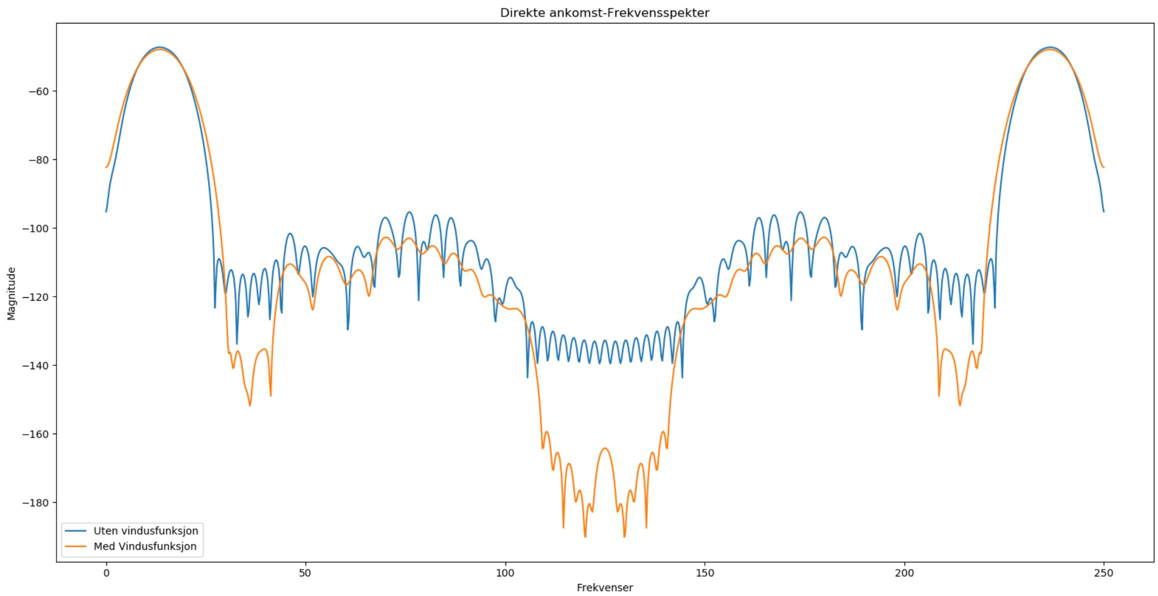


b. Vi skriver videre på programmet, og får:

```
from scipy import signal

w = signal.tukey(len(p), 0.25)
wp = w*p

fwp,fw = frekspekin3190(wp, N, fs)
fp,f = frekspekin3190(p, N, fs)

plt.plot(f, 20*np.log10(abs(fp)))
plt.plot(fw, 20*np.log10(abs(fwp)))
plt.title('Direkte ankomst-Frekvensspekter')
plt.xlabel("Frekvenser")
plt.ylabel("Magnitude")
```

```python
plt.legend(["Uten vindusfunksjon","Med Vindusfunksjon"])
plt.show()
```

Kjøreeksempel gir



Den dominante frekvensen er $13Hz$.

c. Vi har at:

$$f = \frac{c}{\lambda}$$
$$c = \frac{3000m}{s}$$
$$Differansen = \frac{1}{8}$$
$$Dominante\ frekvensen = 13Hz$$

Vi får:

$$f = \frac{c}{\lambda} \rightarrow \lambda = \frac{c}{f} = \frac{\frac{3000m}{s}}{\frac{13}{s}} = 230.8m$$

Den vertikale oppløsningen er derfor:

$$230.8m \cdot \frac{1}{8} = 28.85m$$

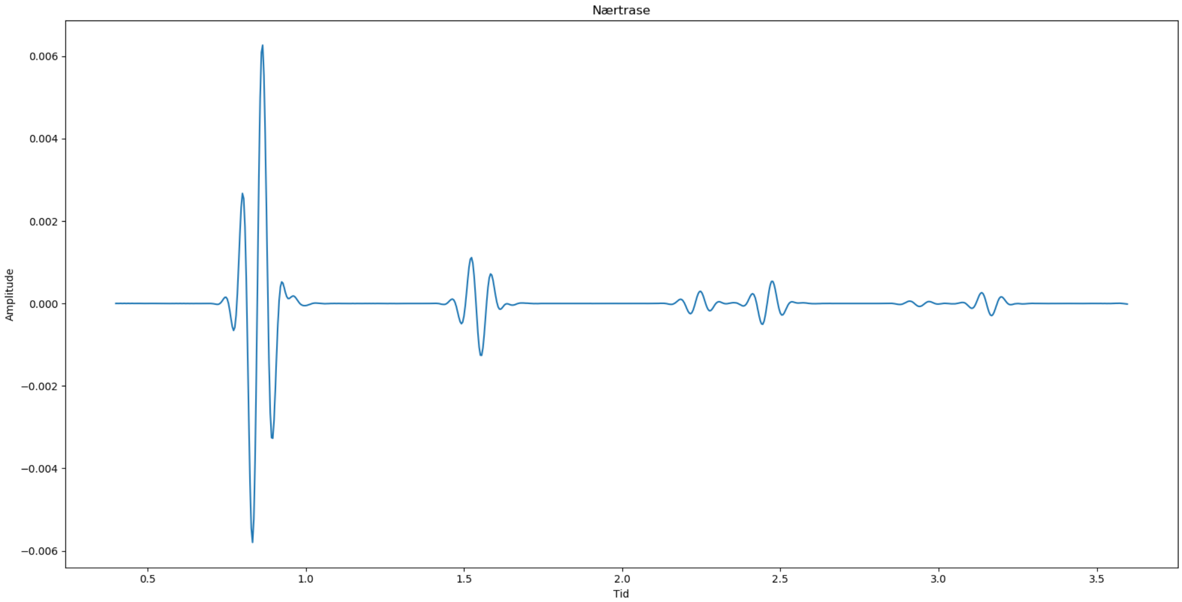
Oppgave 4 (Refleksjoner og multipler)

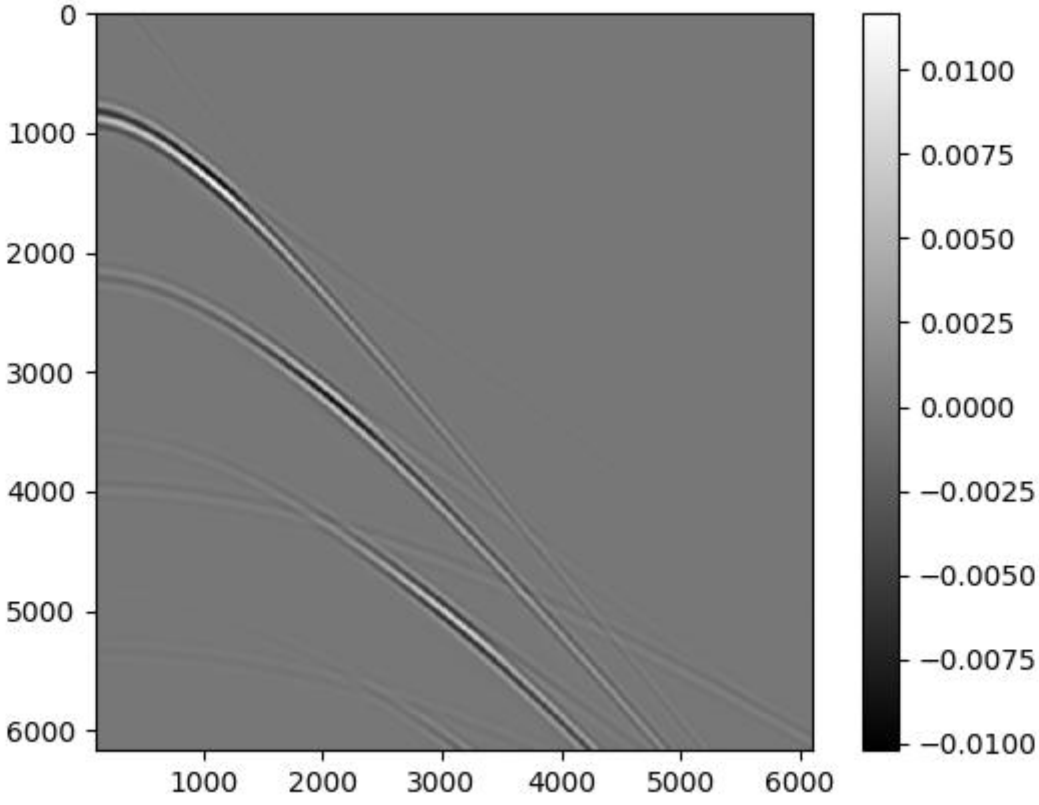
a. Vi skriver videre på programmet fra oppgave 3:

```python
nærtraser1 = y1[100:900,1]
plt.plot(t[100:900], nærtraser1)
plt.title('Nærtrase')
plt.xlabel("Tid")
```

```
plt.ylabel("Amplitude")
plt.show()

plt.imshow(y1[100:900],cmap='gray',extent=[np.amin(offset1), np.amax(offset1),
                                            np.amax(t[900])*1714, 0])
plt.colorbar()
plt.show()
```

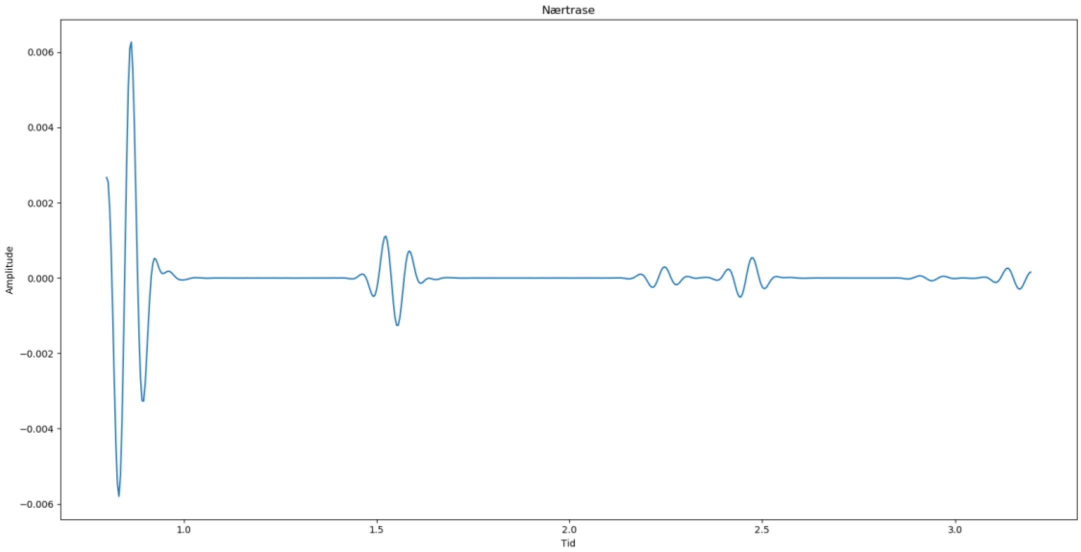
Kjøreeksempel gir:

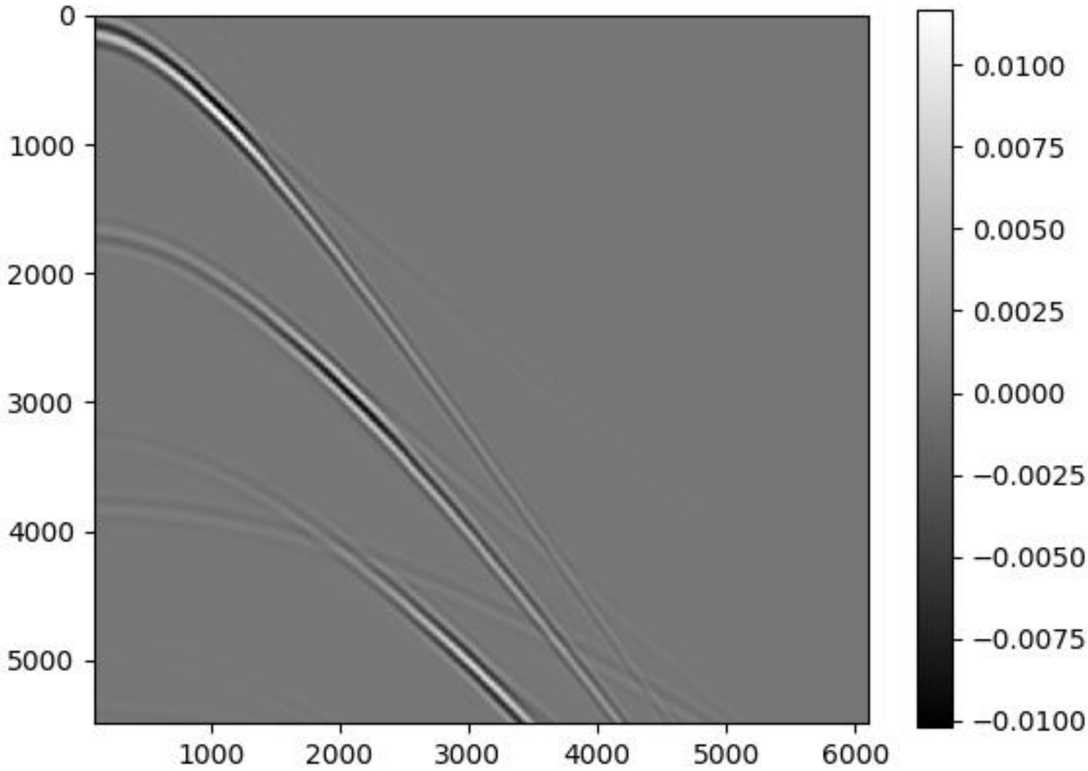




b. Vi skriver videre på samme koden:

```
nærtraser2 = y1[200:800,1]
plt.plot(t[200:800], nærtraser2)
plt.title('Nærtrase')
plt.xlabel("Tid")
plt.ylabel("Amplitude")
plt.show()

plt.imshow(y1[200:800],cmap='gray',extent=[np.amin(offset1),np.amax(offset1),
                                            np.amax(t[800])*1714.29,0])
plt.colorbar()
plt.show()
```





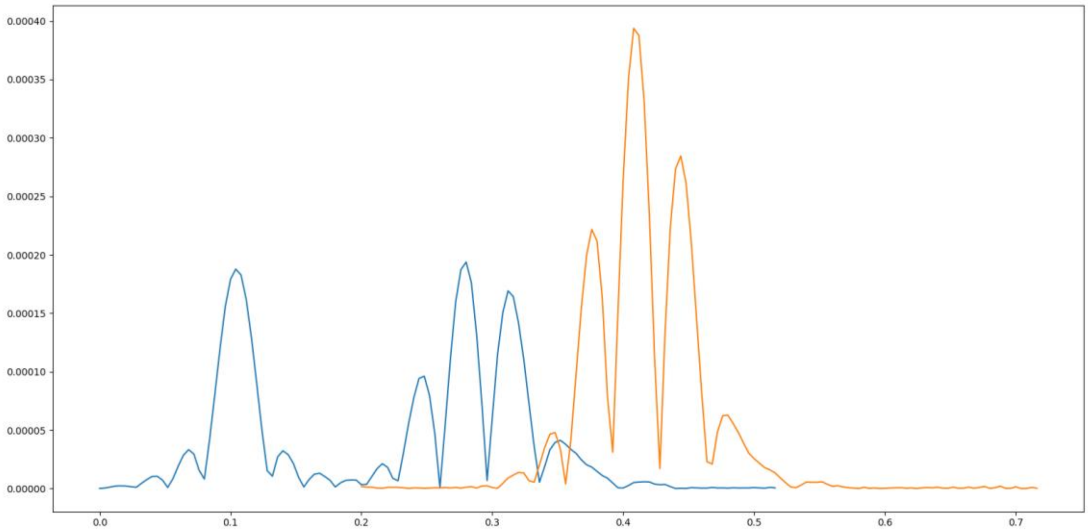
Oppgave 5 (Vannhastighet)

Vi skriver videre på forrige koden:

```
p1 = y1[0:130,10]
p2 = y1[50:180,30]
```

```

#Offset1 = 190, Offset2 = 390

plt.plot(t[0:130],abs(p1))
plt.plot(t[50:180],abs(p2))
plt.show()
```

Og får:



Formel for hastighet er gitt som:

$$v = \frac{\Delta s}{\Delta t} = \frac{390 - 190}{0.41 - 0.29} = \frac{200}{0.12} = \frac{1666.67m}{s}$$

Anslag for hastighet i vannet er da :

$$v = \frac{1666.67m}{s}$$

Oppgave 6 (Sedimenthastighet I)

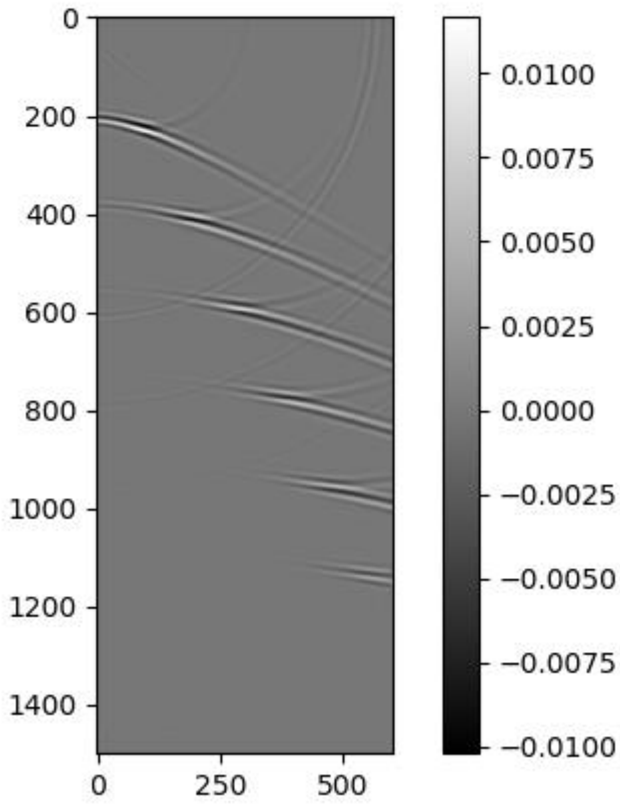
a. Vi skriver programmet i Python

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
import scipy.interpolate as interpolate

#Oppgave6a
klaudiap = sio.loadmat('klaudiap.mat')

t = klaudiap['t']
seismogram1 = klaudiap['seismogram1']
seismogram2 = klaudiap['seismogram2']
offset1 = klaudiap['offset1']
```

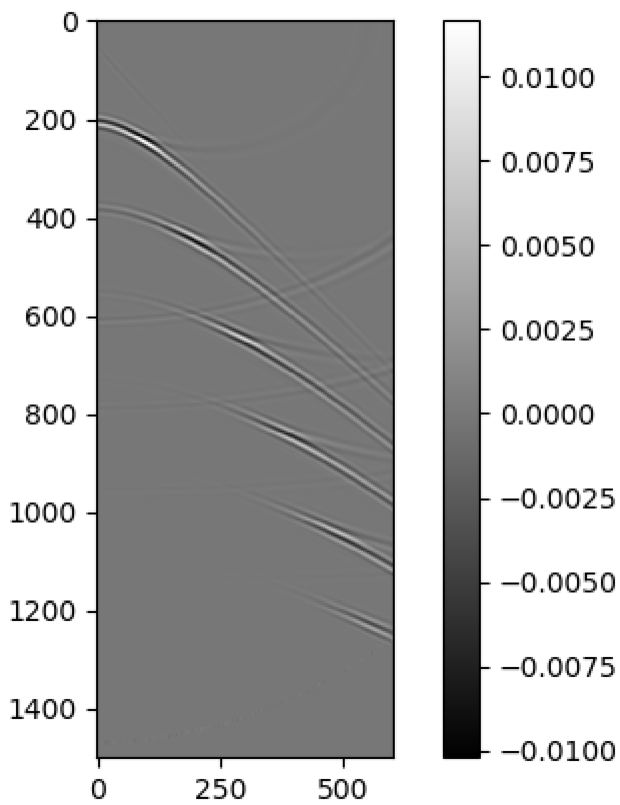
```
offset2 = klaudiap['offset2']

h1 = [0.0002, 0.0001, -0.0001, -0.0005, -0.0011, -0.0017, -0.0019,
      -0.0016, -0.0005, 0.0015, 0.0040, 0.0064, 0.0079, 0.0075, 0.0046,
      -0.0009, -0.0084, -0.0164, -0.0227, -0.0248, -0.0203, -0.0079,
      0.0127, 0.0400, 0.0712, 0.1021, 0.1284, 0.1461, 0.1523, 0.1461,
      0.1284, 0.1021, 0.0712, 0.0400, 0.0127, -0.0079, -0.0203, -0.0248,
      -0.0227, -0.0164, -0.0084, -0.0009, 0.0046, 0.0075, 0.0079, 0.0064,
      0.0040, 0.0015, -0.0005, -0.0016, -0.0019, -0.0017, -0.0011,
      -0.0005, -0.0001, 0.0001, 0.0002]

def nmocorrection(t,dt,offset,seisdata,vnmo):

    """
    nmocorrection NMO correction of CMP gather

    sesnmo = nmo_correction(t,dt,offset,seisdata,vnmo) applies NMO
    correction to a CMP gather according the provided NMO velocities

    Input:
    -t:        Vector containing the travel times of the seismic data (in secounds)
    -dt:       Number containing the time sampling of the seismic data (in secounds)
    -offset:   Vector containing the cource receiver distance for each sesmic
               trace (in  meters)
    -seisdata: Matrix containing columns of seismic traces.
    -vnmo:     Vector containing NMO velocities (in meters/secounds)

    Output:
    -seisnmo:  Matrix containing columns of NMO corrected seismic traces.
    """

    seisnmo = np.zeros(seisdata.shape)
    reflecttimes = np.sqrt(np.square(t) + np.divide(np.square(offset.astype("double")),np.square(vnmo)))
    reflecttimes_1 = reflecttimes.copy()
    reflecttimes = np.multiply(reflecttimes, np.logical_and(reflecttimes >= dt, reflecttimes <= (t[-1] - dt)))
    reflecttimes_2 = reflecttimes.copy()
    for i in range(0, offset.size):
        interp = interpolate.interp1d(np.squeeze(t), seisdata[:,i], axis=0, kind="cubic", bounds_error=False, fill_value=0)
        seisnmo[:,i] = np.squeeze(interp(reflecttimes[:,i]))
    return seisnmo, reflecttimes_1, reflecttimes_2

def konvin3190(h,x,ylen):
    x_len = len(x)
    h_len = len(h)

    if ylen == 1:
```

```
        y = np.zeros(x_len+y_len-1)
        for i in range(h_len):
            for j in range(x_len):
                k = j+i-1
                y[k] = y[k]+h[i]*x[j]

    if ylen == 0:
        y = np.zeros(x_len)
        for i in range(h_len):
            for j in range(x_len-h_len+1):
                k=i+j-1
                y[k] = y[k]+h[i]*x[j]
    return y

y1 = np.zeros_like(seismogram1)
m,n = np.shape(seismogram1)

for i in range(n):
    y1[:,i] = konvin3190(h1,seismogram1[:,i],0)

fs = 1/(t[2]-t[1])
dt = 1/fs

hastighet = 2000

vnmo = np.full((1,601),hastighet)

seisnmo,r1,r2 = nmocorrection(t,dt,offset1,y1,vnmo)

plt.imshow(seisnmo,cmap='gray')
plt.colorbar()
plt.show()
```

Og får ved $v = \frac{1667.67m}{s}$:



Og får ved $v = \frac{2000m}{s}$:

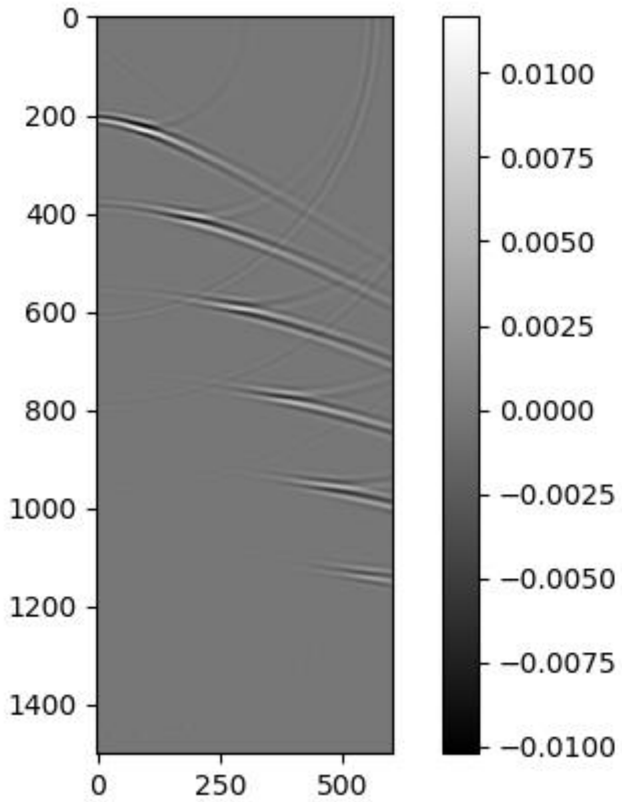


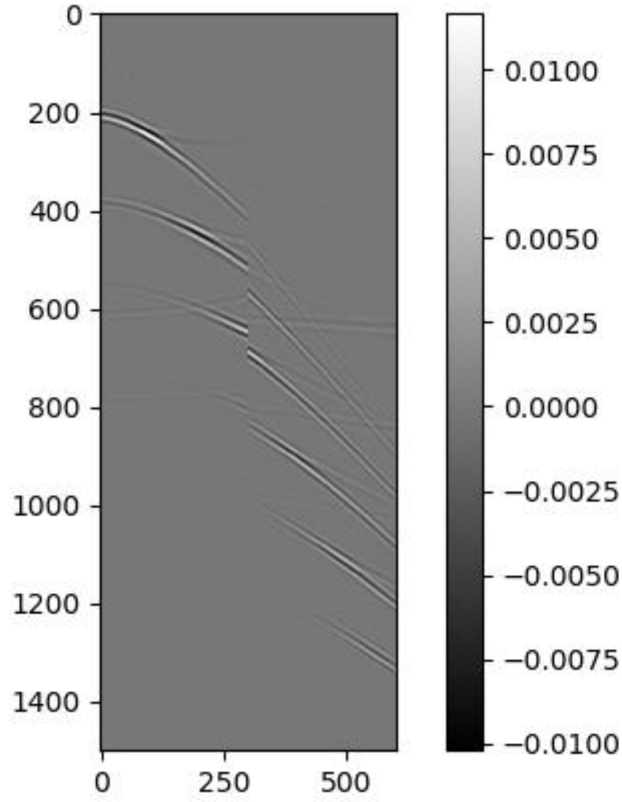
Vi ser at ved $v = \frac{2000m}{s}$ for vi et flatere havbunnsrefleksjon, og vi bruker det derfor i videre utregning.

b. Vi setter nå vektoren vnmo lik hastighet for vannet for tiden det tar for pulsen å bevege seg i vannet, og setter en valgt hastighet for resten. Koden vi får er

```
vnmo2 = np.full((1,601),2500)
for i in range(300):
vnmo2[0,i] = 2000
seisnmo2,r12,r22 = nmocorrection(t,dt,offset1,y1,vnmo2)

plt.imshow(seisnmo2,cmap='gray')
plt.colorbar()
plt.show()
```

Kjøreeksempel gir:





Oppgave 7 (Sedimenthastighet II)

a. Vi ser på plottet vi fikk i oppgave 4a og får:

$$t_1 = 900s$$
$$t_2 = 2100s$$
$$x_1 = 5100$$
$$x_2 = 4900$$

Formel for hastighet er gitt som:

$$v = \frac{\Delta s}{\Delta t} = \frac{5100 - 4900}{2100 - 900} = \frac{200}{1200} = \frac{0.17m}{s}$$

b. Vi ser på plottet vi fikk i oppgave 4b og får:

$$t_1 = 100s$$
$$t_2 = 1800s$$
$$x_1 = 4500$$
$$x_2 = 4100$$

Formel for hastighet er gitt som:

$$v = \frac{\Delta s}{\Delta t} = \frac{4500 - 4100}{1800 - 100} = \frac{400}{1700} = \frac{0.24m}{s}$$

c. Vi skriver videre på programmet og får:

```
m,n = shape(seismogram2)

y2 = np.zeros_like(seismogram2)

```

```python
for i in range(n):
    y2[:,i] = konvin3190(seismogram2[:,i],h1,0)

plt.imshow(y2)
plt.colorbar()
plt.show()
```

Koden virker ikke.

d. Vi ser på figur 2 fra oppgaveteksten. Vi ser at refleksjoner danner rettvinklet trekanter. Vi bruker dem til å regne ut resultatene. Vi bruker hastigheten:

$$v = \frac{2000m}{s}$$

Og finner tiden

$$\Delta t$$

Da får vi dybden:

$$s = v\Delta t$$

Resultatene blir (disse er ikke korrekte):

Beskrivelse	Verdi
Dybde sedimentærlag 1 (i meter)	3922.36
Dybde sedimentærlag 2 (i meter)	586.96
Hastighet vannlag (i m/s)	2000.00
Hastighet sedimentærlag 1 (i m/s)	1177.38
Hastighet sedimentærlag 2 (i m/s)	1960.66