

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:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 #Oppgave 1A
5 def konvin3190(h,x,ylen):
6
7     """
8     konvin3190 konvolverer to signaler
9     konvin3190(h,x,ylen) konvolverer de to vektor-signalene h og x.
10    Dersom ylen = 0 er utgangssignalet lengde "len(x)", mens hvis ylen
11    er 1 har utgangssignalet lengde "len(h)+len(x)-1"
12    """
13
14    x_len = len(x)
15    h_len = len(h)
16
17    if ylen == 1:
18        y = np.zeros(x_len+h_len-1)
19        for i in range(h_len):
20            for j in range(x_len):
21                k = j+i-1
22                y[k] = y[k]+h[i]*x[j]
23
24    if ylen == 0:
25        y = np.zeros(x_len)
26        for i in range(h_len):
27            for j in range(x_len-h_len+1):
28                k=i+j-1
29                y[k] = y[k]+h[i]*x[j]
30    return y
```

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

```
1 #Oppgave 1B
2 def frekspekin3190(x,N,fs):
3
4     """
```

```

5  frekspekin3190 regner ut frekvensresponsen til et signal
6  x,g = frekspekin3190(x,N,fs) regner ut frekvensresponsen til
7  signalet x med samplingfrekvens fs for N punkter på enhetssirkelen
8  I tillegg til frekvensspekteret X returnerer funksjonen også tilhørende
9  frekvens f
10  """
11
12  x_len = len(x)
13  X = np.zeros((N),dtype = "complex_")
14  fp = np.linspace(0,1,N)
15  f = fp*fs
16
17  for i in range(N):
18      for j in range(x_len):
19          X[i] = X[i]+x[j]*np.exp(2*np.pi*-1j*fp[i]*j)
20
21  return X,f

```

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

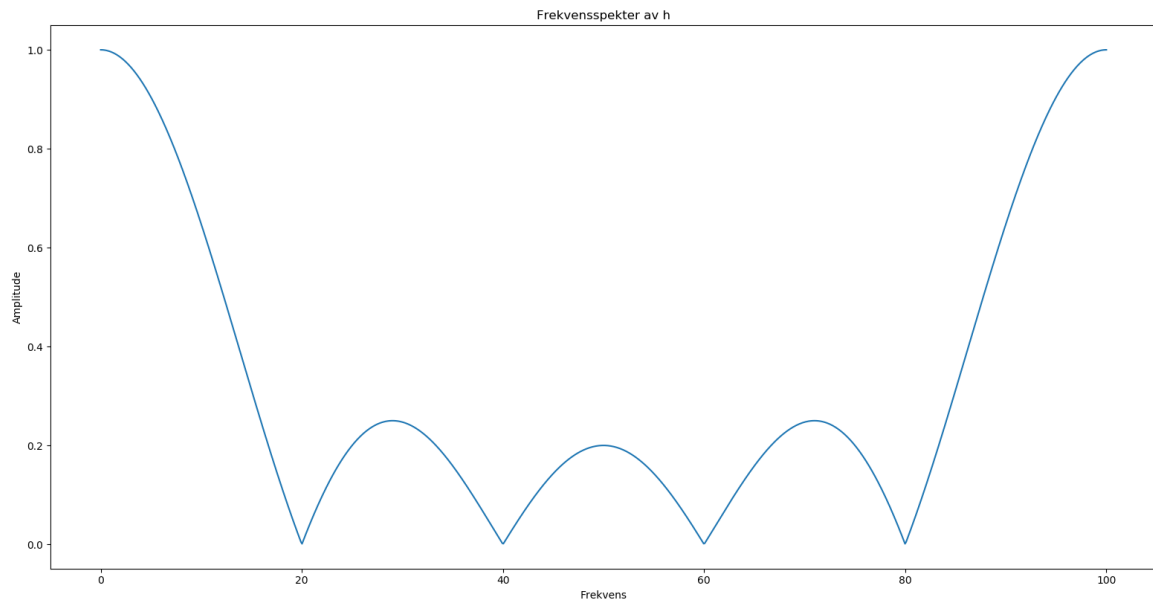
```

1  #Oppgave 1C
2  f1 = 10
3  f2 = 20
4  fs = 100
5  t_len = 5
6
7  t = np.linspace(0,t_len,t_len*fs+1)
8  h = [0.2,0.2,0.2,0.2,0.2] #FIR filter
9  N = 1000 #Vi velger antall punkter
10 x = np.sin(2*np.pi*f1*t)+np.sin(2*np.pi*f2*t) #Den gitte signalet
11
12 Xh,fh = frekspekin3190(h,N,fs) #Frekvensspekter av h
13 Xx,fx = frekspekin3190(x,N,fs) #Frekvensspekter av x
14 Xy,fy = frekspekin3190(konvin3190(h,x,0),N,fs) #Frekvensspekter av y
15
16 plt.plot(fh,abs(Xh))
17 plt.title('Frekvensspekter av h')
18 plt.xlabel('Frekvens')
19 plt.ylabel('Amplitude')
20 plt.show()
21
22 plt.subplot(211);
23 plt.plot(fy,abs(Xy))
24 plt.title('Frekvensspekter av y')
25 plt.xlabel('Frekvens')
26 plt.ylabel('Amplitude')
27
28 plt.subplot(212);
29 plt.plot(fx,abs(Xx))

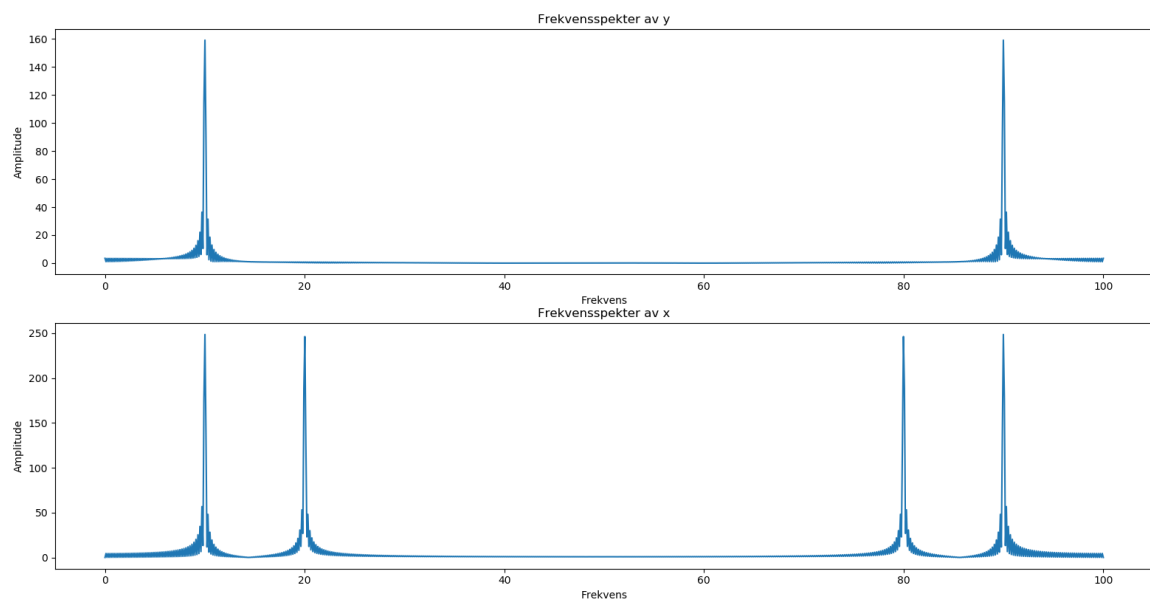
```

```
30 plt.title('Frekvensspekter av x')
31 plt.xlabel('Frekvens')
32 plt.ylabel('Amplitude')
33 plt.show()
```

Og vi får:



Og:



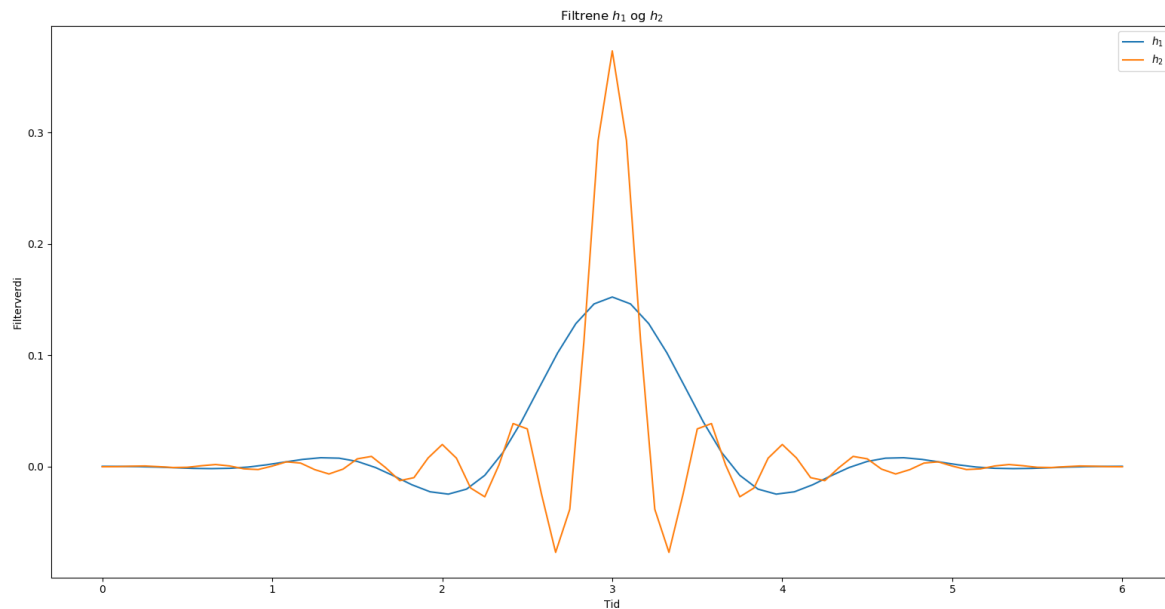
Vi får en topp ved 10Hz på frekvensspekter av y , og to topper 10Hz og 20Hz på frekvensspekter av x (de andre toppene er en speiling). Vi ser at toppen på 20Hz ble filtrert ut (pga. konvolusjon), samtidig som amplituden har blitt mindre. Dette stemmer med frekvensspekter av h , da amplituden er 0 ved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

Oppgave 2 (Støyfjerning)

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

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.io as sio
4
5 #Oppgave2A
6 kaudiap = sio.loadmat('kaudiap.mat')
7
8 t = kaudiap['t']
9 seismogram1 = kaudiap['seismogram1']
10 seismogram2 = kaudiap['seismogram2']
11 offset1 = kaudiap['offset1']
12 offset2 = kaudiap['offset2']
13
14 h1 = [0.0002, 0.0001, -0.0001, -0.0005, -0.0011, -0.0017, -0.0019,
15       -0.0016, -0.0005, 0.0015, 0.0040, 0.0064, 0.0079, 0.0075, 0.0046,
16       -0.0009, -0.0084, -0.0164, -0.0227, -0.0248, -0.0203, -0.0079,
17       0.0127, 0.0400, 0.0712, 0.1021, 0.1284, 0.1461, 0.1523, 0.1461,
18       0.1284, 0.1021, 0.0712, 0.0400, 0.0127, -0.0079, -0.0203, -0.0248,
19       -0.0227, -0.0164, -0.0084, -0.0009, 0.0046, 0.0075, 0.0079, 0.0064,
20       0.0040, 0.0015, -0.0005, -0.0016, -0.0019, -0.0017, -0.0011,
21       -0.0005, -0.0001, 0.0001, 0.0002]
22
23 h2 = [-0.0002, -0.0001, 0.0003, 0.0005, -0.0001, -0.0009, -0.0007,
24       0.0007, 0.0018, 0.0005, -0.0021, -0.0027, 0.0004, 0.0042, 0.0031,
25       -0.0028, -0.0067, -0.0023, 0.0069, 0.0091, -0.0010, -0.0127,
26       -0.0100, 0.0077, 0.0198, 0.0075, -0.0193, -0.0272, 0.0014, 0.0386,
27       0.0338, -0.0246, -0.0771, -0.0384, 0.1128, 0.2929, 0.3734, 0.2929,
28       0.1128, -0.0384, -0.0771, -0.0246, 0.0338, 0.0386, 0.0014, -0.0272,
29       -0.0193, 0.0075, 0.0198, 0.0077, -0.0100, -0.0127, -0.0010, 0.0091,
30       0.0069, -0.0023, -0.0067, -0.0028, 0.0031, 0.0042, 0.0004, -0.0027,
31       -0.0021, 0.0005, 0.0018, 0.0007, -0.0007, -0.0009, -0.0001, 0.0005,
32       0.0003, -0.0001, -0.0002]
33
34 t1 = np.linspace(np.amin(t), np.amax(t), len(h1))
35 t2 = np.linspace(np.amin(t), np.amax(t), len(h2))
36
37 plt.plot(t1, h1)
38 plt.plot(t2, h2)
39 plt.title('Filtrene $h_1$ og $h_2$')
40 plt.xlabel('Tid')
41 plt.ylabel('Filterverdi')
42 plt.legend(['$h_1$', '$h_2$'])
43 plt.show()
```

Og får:



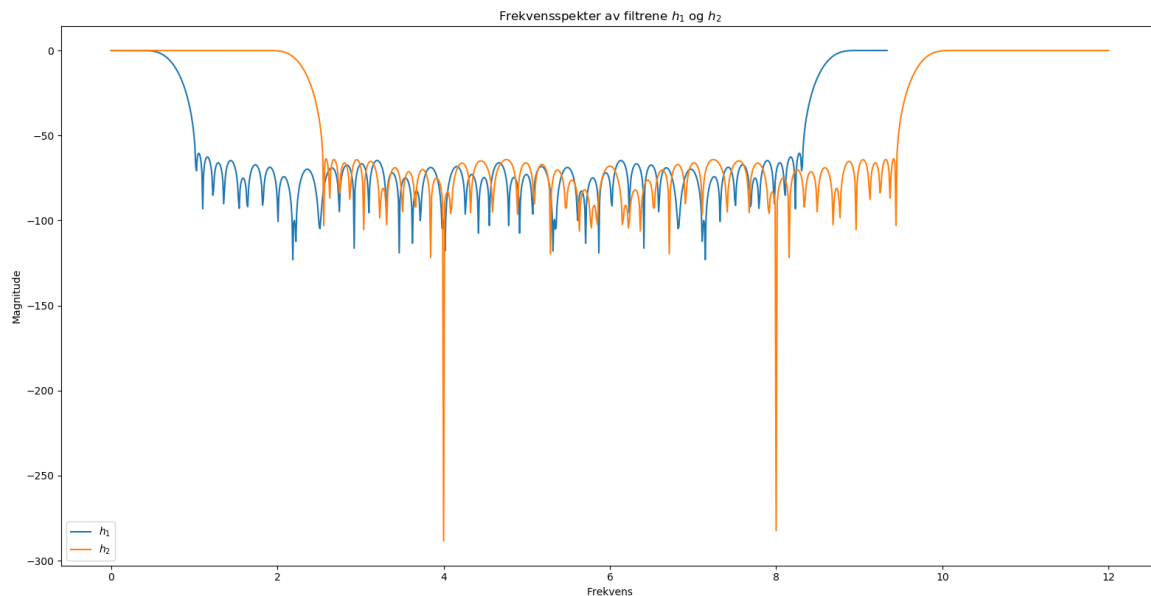
Vi skriver videre på programmet for å regne ut frekvensspekteret til h_1 og h_2 :

```

44 def frekspekin3190(x,N,fs):
45     x_len = len(x)
46     X = np.zeros((N),dtype = "complex_")
47     fp = np.linspace(0,1,N)
48     f = fp*fs
49
50     for i in range(N):
51         for j in range(x_len):
52             X[i] = X[i]+x[j]*np.exp(2*np.pi*-1j*fp[i]*j)
53
54     return X,f
55
56 N = 1000
57 fs1 = 1/(t1[2]-t1[1])
58 fs2 = 1/(t2[2]-t2[1])
59
60 H1,fh1 = frekspekin3190(h1,N,fs1)
61 H2,fh2 = frekspekin3190(h2,N,fs2)
62
63 plt.plot(fh1,20*np.log10(abs(H1)),fh2,20*np.log10(abs(H2)))
64 plt.title('Frekvensspekter av filtrene $h_1$ og $h_2$')
65 plt.xlabel('Frekvens')
66 plt.ylabel('Magnitude')
67 plt.legend(["$h_1$", "$h_2$"])
68 plt.show()

```

Og får

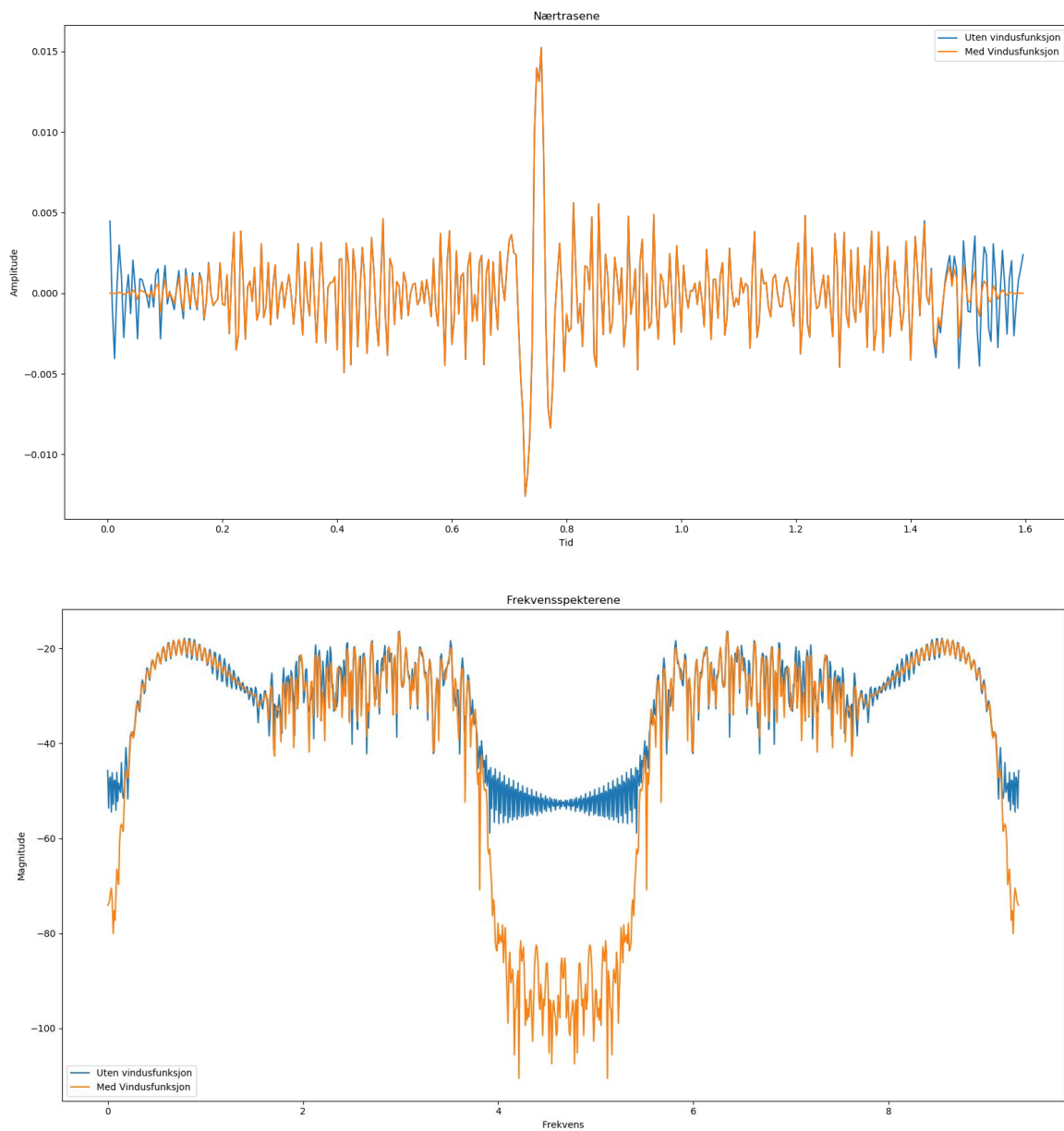


Vi ser filter h_1 kan filtrere mer frekvenser enn h_2 . h_1 filtrerer de lave frekvenser (fra 1Hz til 2.5Hz), men ikke de høye frekvenser (8.2Hz til 9.5Hz), og h_2 filtrerer de høye frekvenser (8.2Hz til 9.5Hz), men ikke de lave frekvenser (fra 1Hz til 2.5Hz).

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

```
69 from scipy import signal
70
71 s = seismogram1[1:400,1]
72 w = signal.tukey(len(s),0.25)
73
74 sw = s*w
75
76 plt.plot(t[1:400], s)
77 plt.plot(t[1:400], sw)
78 plt.title('Nærtrasene')
79 plt.xlabel('Tid')
80 plt.ylabel('Amplitude')
81 plt.legend(["Uten vindusfunksjon", "Med Vindusfunksjon"])
82 plt.show()
83
84 fs,f = frekspekin3190(s,N,fs1)
85 fsw,fw = frekspekin3190(sw,N,fs1)
86
87 plt.plot(f,20*np.log10(abs(fs)))
88 plt.plot(fw,20*np.log10(abs(fsw)))
89 plt.title('Frekvensspekterene')
90 plt.xlabel('Frekvens')
91 plt.ylabel('Magnitude')
92 plt.legend(["Uten vindusfunksjon", "Med Vindusfunksjon"])
93 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:

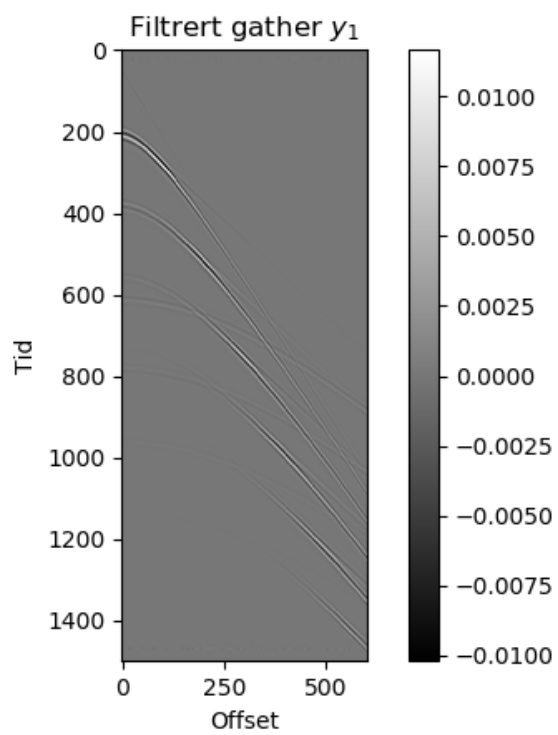
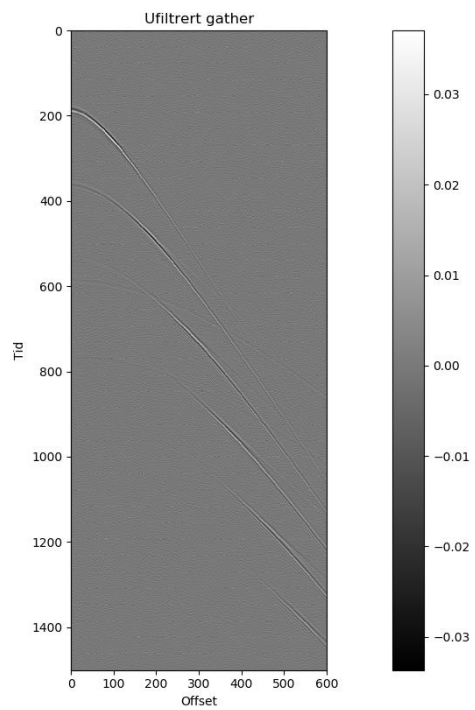
```
1 def konvin3190(h,x,ylen):
2     x_len = len(x)
3     h_len = len(h)
4
5     if ylen == 1:
6         y = np.zeros(x_len+y_len-1)
7         for i in range(h_len):
8             for j in range(x_len):
```

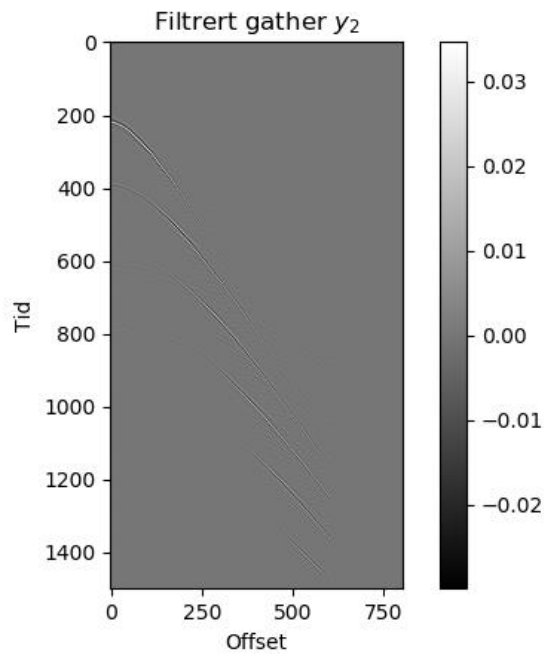
```

9         k = j+i-1
10        y[k] = y[k]+h[i]*x[j]
11
12    if ylen == 0:
13        y = np.zeros(x_len)
14        for i in range(h_len):
15            for j in range(x_len-h_len+1):
16                k=i+j-1
17                y[k] = y[k]+h[i]*x[j]
18    return y
19
20 y1 = np.zeros_like(seismogram1)
21 y2 = np.zeros_like(seismogram2)
22 m,n = np.shape(seismogram1)
23
24 for i in range(n):
25     y1[:,i] = konvin3190(h1,seismogram1[:,i],0)
26     y2[:,i] = konvin3190(h2,seismogram1[:,i],0)
27
28 plt.imshow(seismogram1,cmap='gray')
29 plt.title('Ufiltrert gather')
30 plt.xlabel('Offset')
31 plt.ylabel('Tid')
32 plt.colorbar()
33 plt.show()
34
35 plt.imshow(y1,cmap='gray')
36 plt.title('Filtrert gather $y_1$')
37 plt.xlabel('Offset')
38 plt.ylabel('Tid')
39 plt.colorbar()
40 plt.show()
41
42 plt.imshow(y2,cmap='gray')
43 plt.title('Filtrert gather $y_2$')
44 plt.xlabel('Offset')
45 plt.ylabel('Tid')
46 plt.colorbar()
47 plt.show()

```

Kjøreeksempel gir:

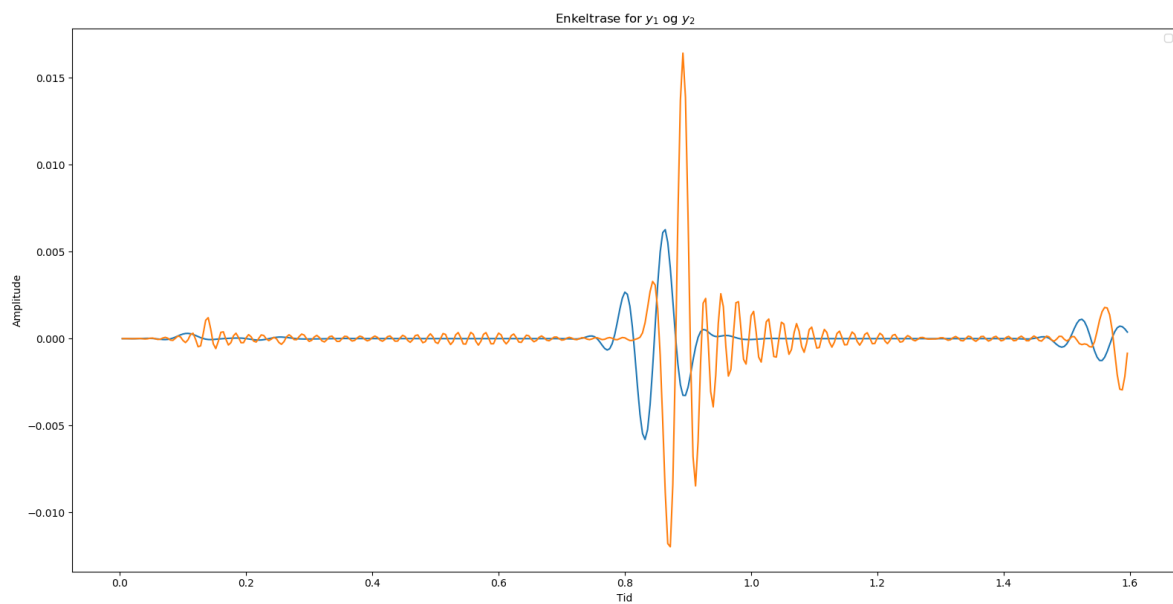




Vi fortsetter på programmet:

```
1 plt.plot(t[1:400], y1[1:400,1],t[1:400],y2[1:400,1])
2 plt.title('Enkeltrase for $y_1$ og $y_2$')
3 plt.xlabel('Tid')
4 plt.ylabel('Amplitude')
5 plt.legend('$y_1$', '$y_2$')
6 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:

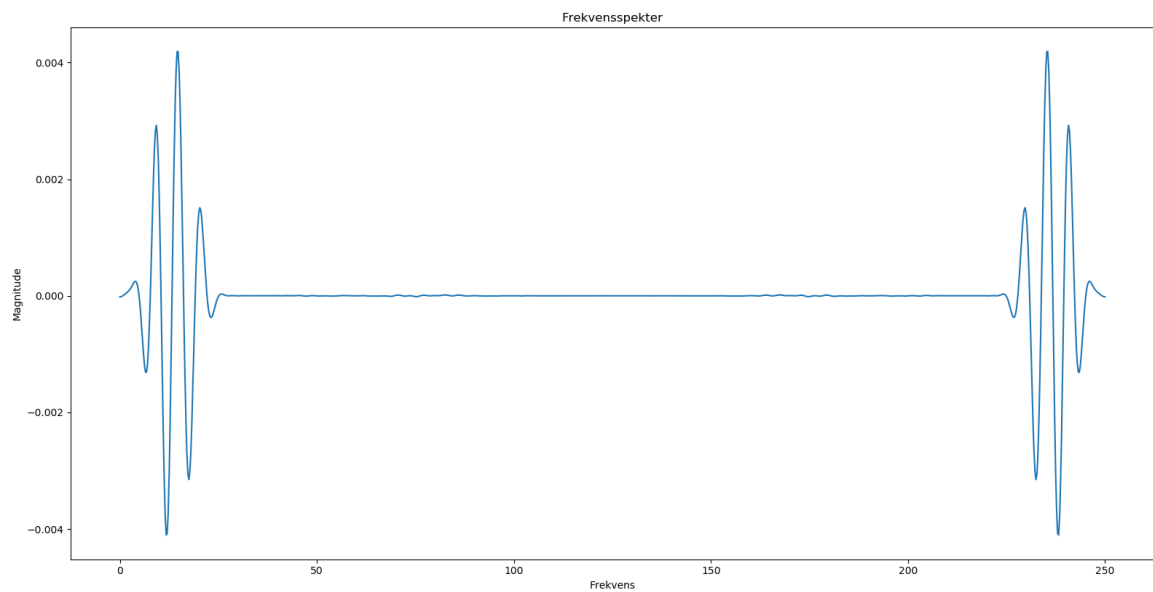
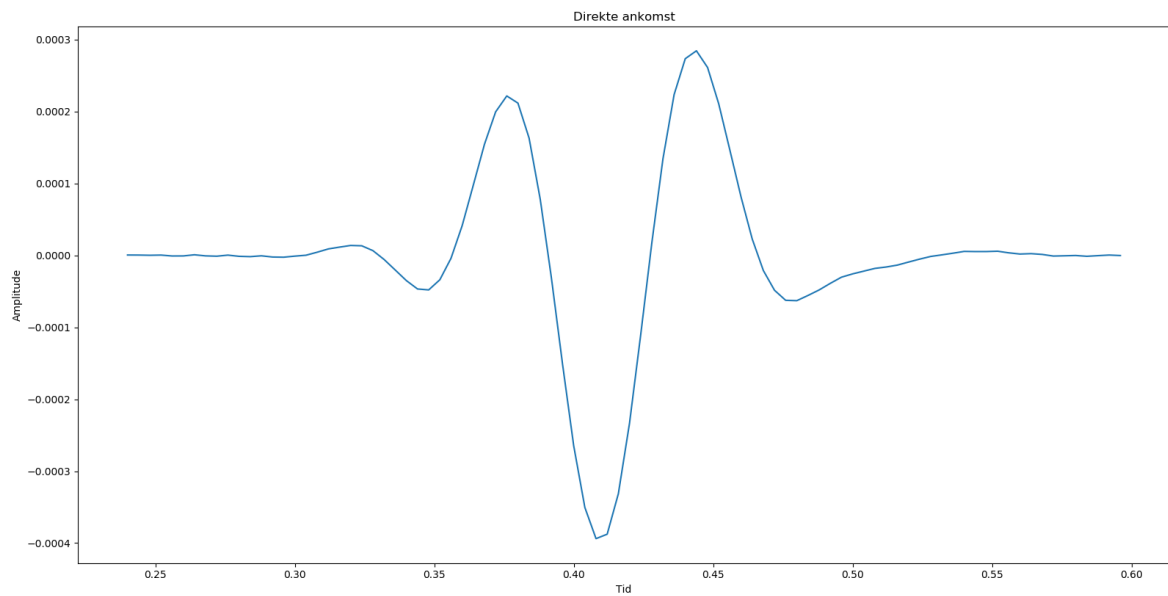
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.io as sio
4
5 kaudiap = sio.loadmat('klaudiap.mat')
6
7 t = kaudiap['t']
8 seismogram1 = kaudiap['seismogram1']
9 seismogram2 = kaudiap['seismogram2']
10 offset1 = kaudiap['offset1']
11 offset2 = kaudiap['offset2']
12
13 h1 = [0.0002, 0.0001, -0.0001, -0.0005, -0.0011, -0.0017, -0.0019,
14       -0.0016, -0.0005, 0.0015, 0.0040, 0.0064, 0.0079, 0.0075, 0.0046,
15       -0.0009, -0.0084, -0.0164, -0.0227, -0.0248, -0.0203, -0.0079,
16       0.0127, 0.0400, 0.0712, 0.1021, 0.1284, 0.1461, 0.1523, 0.1461,
17       0.1284, 0.1021, 0.0712, 0.0400, 0.0127, -0.0079, -0.0203, -0.0248,
18       -0.0227, -0.0164, -0.0084, -0.0009, 0.0046, 0.0075, 0.0079, 0.0064,
19       0.0040, 0.0015, -0.0005, -0.0016, -0.0019, -0.0017, -0.0011,
20       -0.0005, -0.0001, 0.0001, 0.0002]
21
22 h2 = [-0.0002, -0.0001, 0.0003, 0.0005, -0.0001, -0.0009, -0.0007,
23       0.0007, 0.0018, 0.0005, -0.0021, -0.0027, 0.0004, 0.0042, 0.0031,
24       -0.0028, -0.0067, -0.0023, 0.0069, 0.0091, -0.0010, -0.0127,
25       -0.0100, 0.0077, 0.0198, 0.0075, -0.0193, -0.0272, 0.0014, 0.0386,
26       0.0338, -0.0246, -0.0771, -0.0384, 0.1128, 0.2929, 0.3734, 0.2929,
27       0.1128, -0.0384, -0.0771, -0.0246, 0.0338, 0.0386, 0.0014, -0.0272,
28       -0.0193, 0.0075, 0.0198, 0.0077, -0.0100, -0.0127, -0.0010, 0.0091,
29       0.0069, -0.0023, -0.0067, -0.0028, 0.0031, 0.0042, 0.0004, -0.0027,
30       -0.0021, 0.0005, 0.0018, 0.0007, -0.0007, -0.0009, -0.0001, 0.0005,
31       0.0003, -0.0001, -0.0002]
32
33 t1 = np.linspace(np.amin(t), np.amax(t), len(h1))
34 t2 = np.linspace(np.amin(t), np.amax(t), len(h2))
35
36 def konvin3190(h, x, ylen):
37     x_len = len(x)
38     h_len = len(h)
39
40     if ylen == 1:
41         y = np.zeros(x_len + ylen - 1)
42         for i in range(h_len):
43             for j in range(x_len):
44                 k = j + i - 1
45                 y[k] = y[k] + h[i] * x[j]
```

```

46
47     if ylen == 0:
48         y = np.zeros(x_len)
49         for i in range(h_len):
50             for j in range(x_len-h_len+1):
51                 k=i+j-1
52                 y[k] = y[k]+h[i]*x[j]
53         return y
54
55 def frekspekin3190(x,N,fs):
56     x_len = len(x)
57     X = np.zeros((N),dtype = "complex_")
58     fp = np.linspace(0,1,N)
59     f = fp*fs
60
61     for i in range(N):
62         for j in range(x_len):
63             X[i] = X[i]+x[j]*np.exp(2*np.pi*-1j*fp[i]*j)
64
65     return X,f
66
67 N = 1000
68 fs = 1/(t[2]-t[1])
69 y1 = np.zeros_like(seismogram1)
70 m,n = np.shape(seismogram1)
71
72 for i in range(n):
73     y1[:,i] = konvin3190(h1,seismogram1[:,i],0)
74 p = y1[60:150,30]
75
76 plt.plot(t[60:150],p)
77 plt.title('Direkte ankomst')
78 plt.xlabel("Tid")
79 plt.ylabel("Amplitude")
80 plt.show()
81
82 Sl,sf = frekspekin3190(p, N, fs)
83
84 plt.plot(sf, Sl)
85 plt.title('Frekvensspekter')
86 plt.xlabel("Frekvens")
87 plt.ylabel("Magnitute")
88 plt.show()

```

Kjøreeksempel gir:



b. Vi skriver videre på programmet, og får:

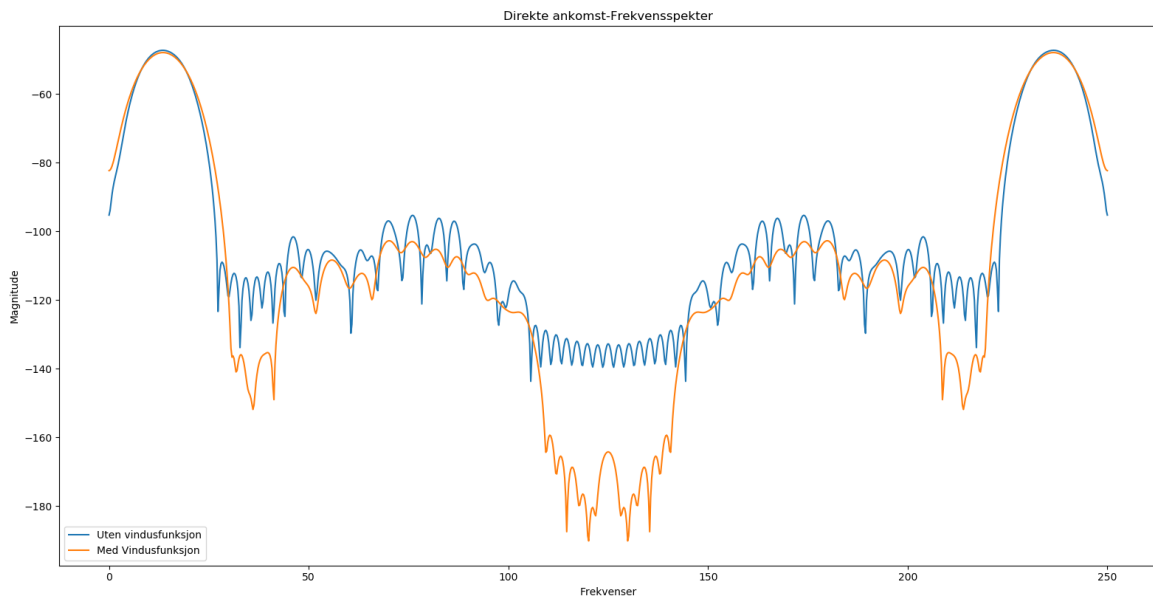
```

89 from scipy import signal
90
91 w = signal.tukey(len(p), 0.25)
92 wp = w*p
93
94 fwp,fw = frekspekin3190(wp, N, fs)
95 fp,f = frekspekin3190(p, N, fs)
96
97 plt.plot(f, 20*np.log10(abs(fp)))
98 plt.plot(fw, 20*np.log10(abs(fwp)))
99 plt.title('Direkte ankomst-Frekvensspekter')
100 plt.xlabel("Frekvenser")
101 plt.ylabel("Magnitude")

```

```
102 plt.legend(["Uten vindusfunksjon", "Med Vindusfunksjon"])
103 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:

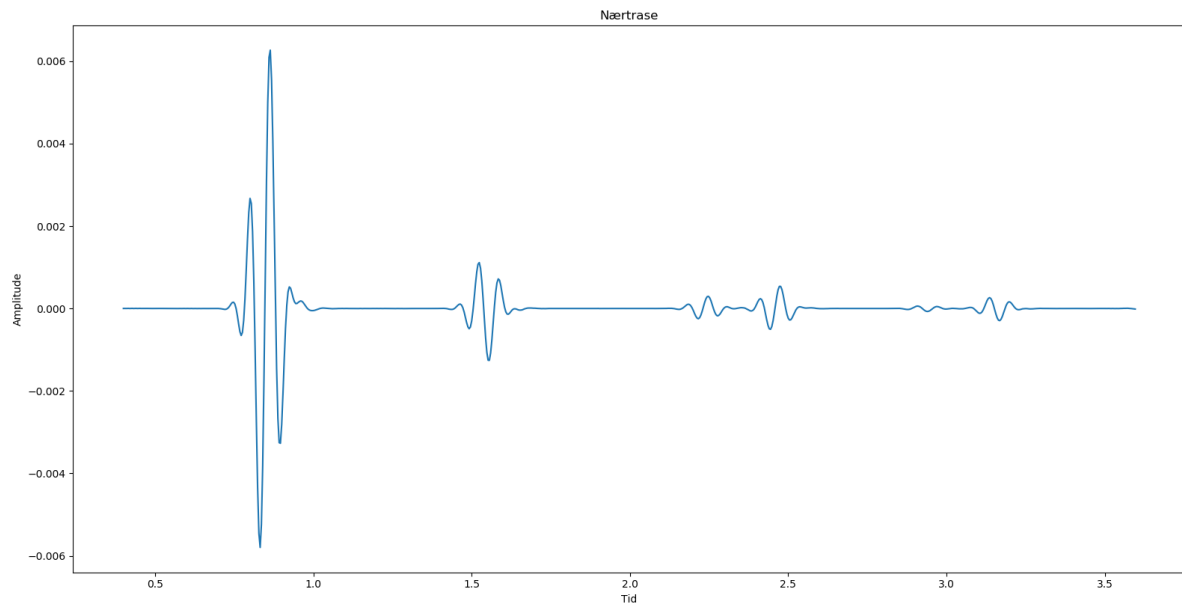
```
1 nærtraser1 = y1[100:900,1]
2 plt.plot(t[100:900], nærtraser1)
3 plt.title('Nærtrase')
4 plt.xlabel("Tid")
```

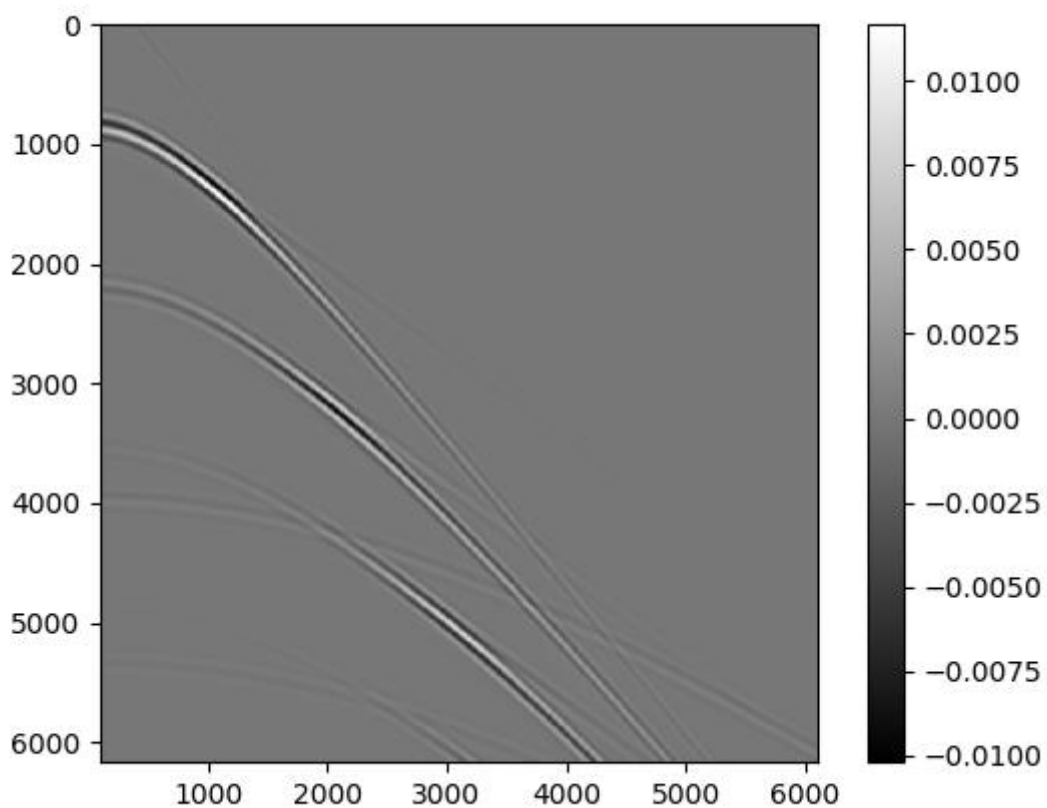
```

5 plt.ylabel("Amplitude")
6 plt.show()
7
8 plt.imshow(y1[100:900],cmap='gray',extent=[np.amin(offset1), np.amax(offset1),
9                                             np.amax(t[900])*1714, 0])
10 plt.colorbar()
11 plt.show()

```

Kjøreeksempel gir:



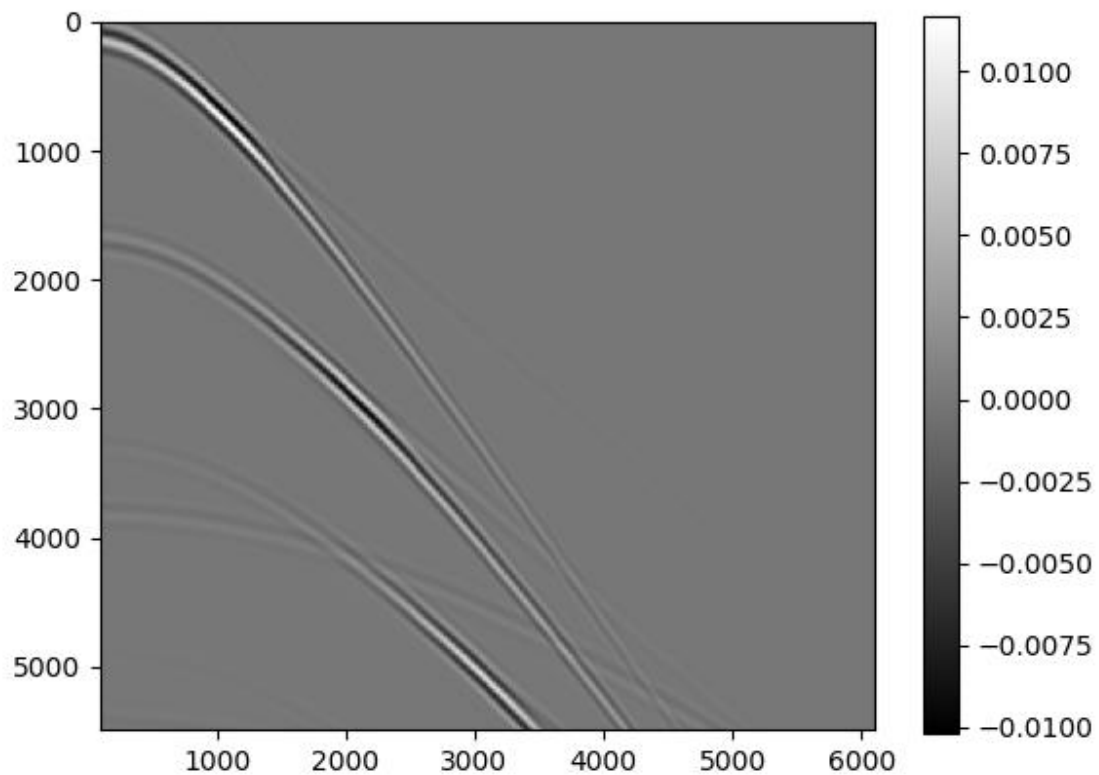
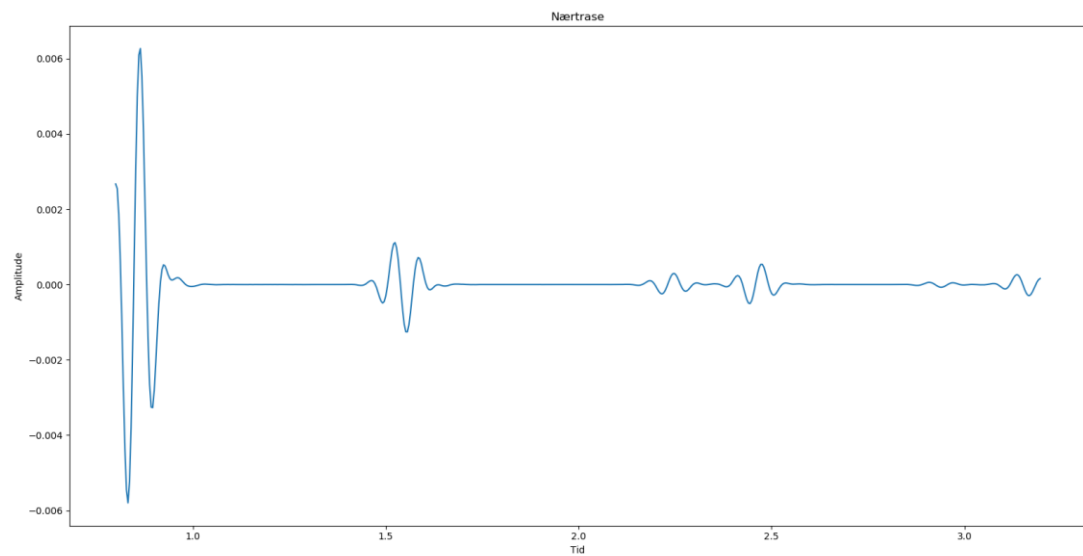


b. Vi skriver videre på samme koden:

```

1  nærtraser2 = y1[200:800,1]
2  plt.plot(t[200:800], nærtraser2)
3  plt.title('Nærtrase')
4  plt.xlabel("Tid")
5  plt.ylabel("Amplitude")
6  plt.show()
7
8  plt.imshow(y1[200:800], cmap='gray', extent=[np.amin(offset1), np.amax(offset1),
9  np.amax(t[800])*1714.29, 0])
10 plt.colorbar()
11 plt.show()

```

Oppgave 5 (Vannhastighet)

Vi skriver videre på forrige koden:

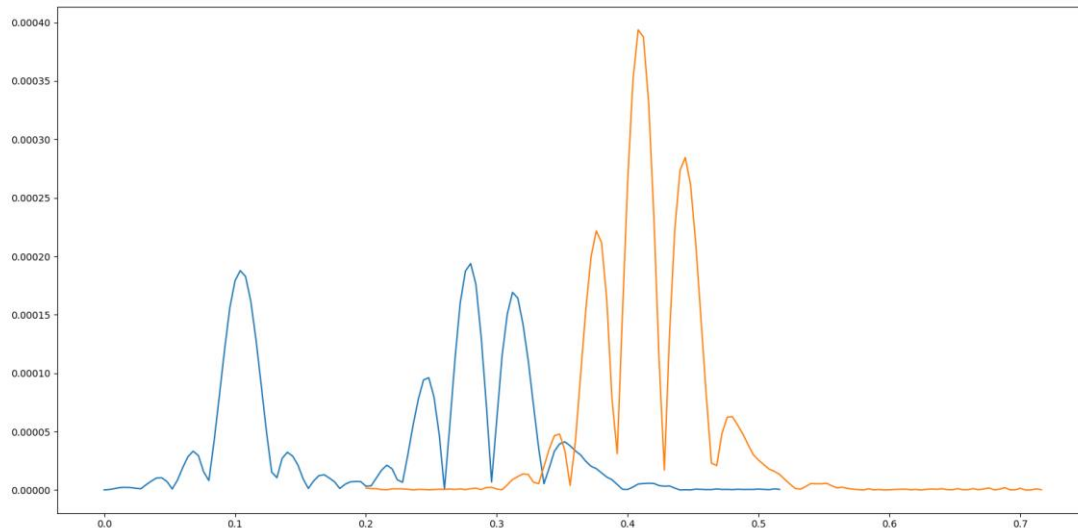
```
1 p1 = y1[0:130,10]
2 p2 = y1[50:180,30]
```

```

3
4 #Offset1 = 190, Offset2 = 390
5
6 plt.plot(t[0:130],abs(p1))
7 plt.plot(t[50:180],abs(p2))
8 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

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.io as sio
4 import scipy.interpolate as interpolate
5
6 #Oppgave6a
7 klaudiap = sio.loadmat('klaudiap.mat')
8
9 t = klaudiap['t']
10 seismogram1 = klaudiap['seismogram1']
11 seismogram2 = klaudiap['seismogram2']
12 offset1 = klaudiap['offset1']

```

```

13 offset2 = klaudiap['offset2']
14
15 h1 = [0.0002, 0.0001, -0.0001, -0.0005, -0.0011, -0.0017, -0.0019,
16       -0.0016, -0.0005, 0.0015, 0.0040, 0.0064, 0.0079, 0.0075, 0.0046,
17       -0.0009, -0.0084, -0.0164, -0.0227, -0.0248, -0.0203, -0.0079,
18       0.0127, 0.0400, 0.0712, 0.1021, 0.1284, 0.1461, 0.1523, 0.1461,
19       0.1284, 0.1021, 0.0712, 0.0400, 0.0127, -0.0079, -0.0203, -0.0248,
20       -0.0227, -0.0164, -0.0084, -0.0009, 0.0046, 0.0075, 0.0079, 0.0064,
21       0.0040, 0.0015, -0.0005, -0.0016, -0.0019, -0.0017, -0.0011,
22       -0.0005, -0.0001, 0.0001, 0.0002]
23
24 def nmocorrection(t,dt,offset,seisdata,vnmo):
25
26     """
27     nmocorrection NMO correction of CMP gather
28
29     sesnmo = nmo_correction(t,dt,offset,seisdata,vnmo) applies NMO
30     correction to a CMP gather according the provided NMO velocities
31
32     Input:
33     -t:      Vector containing the travel times of the seismic data (in seconds)
34     -dt:     Number containing the time sampling of the seismic data (in seconds)
35     -offset: Vector containing the source receiver distance for each seismic
36             trace (in meters)
37     -seisdata: Matrix containing columns of seismic traces.
38     -vnmo:    Vector containing NMO velocities (in meters/seconds)
39
40     Output:
41     -seisnmo: Matrix containing columns of NMO corrected seismic traces.
42     """
43
44     seisnmo = np.zeros(seisdata.shape)
45     reflecttimes = np.sqrt(np.square(t) + np.divide(np.square(offset.astype("double")),np.s
46     quare(vnmo)))
47     reflecttimes_1 = reflecttimes.copy()
48     reflecttimes = np.multiply(reflecttimes, np.logical_and(reflecttimes >= dt, reflecttimes
49     <= (t[-1] - dt)))
50     reflecttimes_2 = reflecttimes.copy()
51     for i in range(0, offset.size):
52         interp = interpolate.interp1d(np.squeeze(t), seisdata[:,i], axis=0, kind="cubic", bound
53         s_error=False, fill_value=0)
54         seisnmo[:,i] = np.squeeze(interp(reflecttimes[:,i]))
55     return seisnmo, reflecttimes_1, reflecttimes_2
56
57 def konvin3190(h,x,ylen):
58     x_len = len(x)
59     h_len = len(h)
60
61     if ylen == 1:

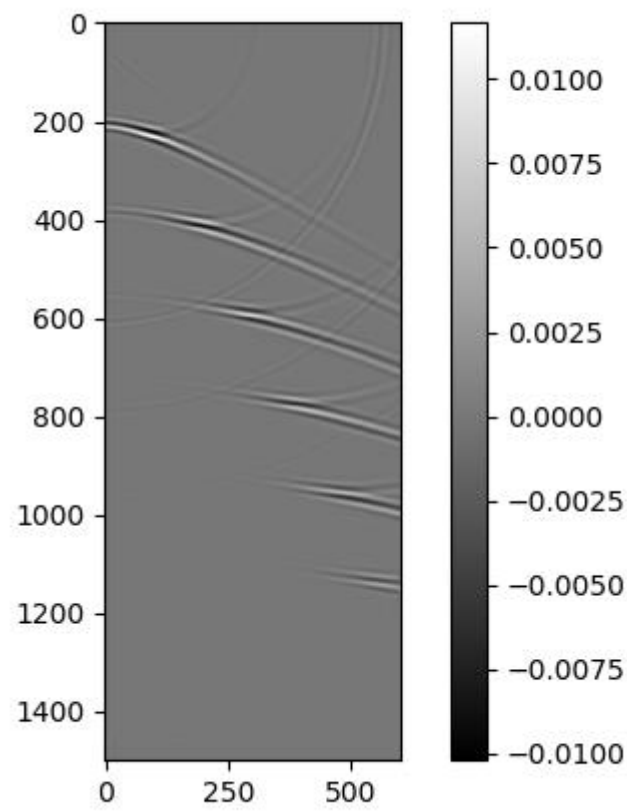
```

```

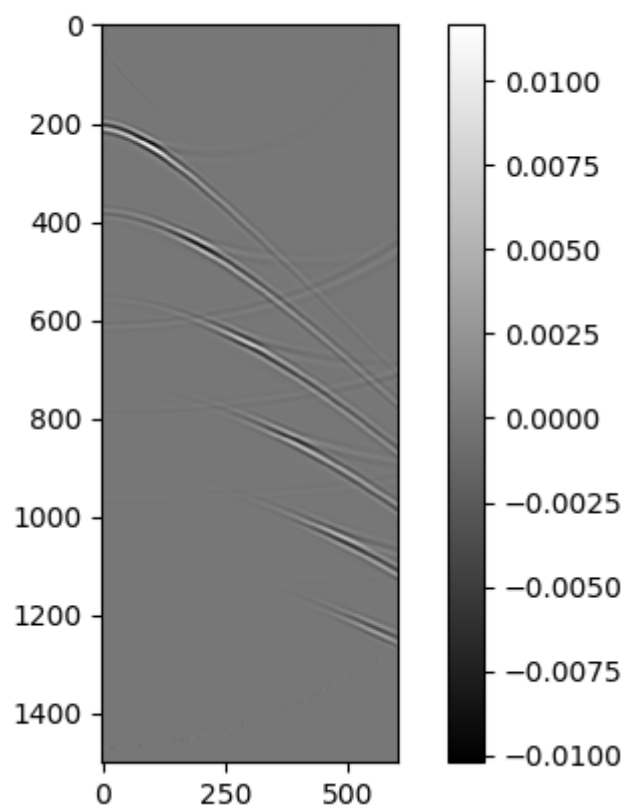
59     y = np.zeros(x_len+y_len-1)
60     for i in range(h_len):
61         for j in range(x_len):
62             k = j+i-1
63             y[k] = y[k]+h[i]*x[j]
64
65     if ylen == 0:
66         y = np.zeros(x_len)
67         for i in range(h_len):
68             for j in range(x_len-h_len+1):
69                 k=i+j-1
70                 y[k] = y[k]+h[i]*x[j]
71     return y
72
73 y1 = np.zeros_like(seismogram1)
74 m,n = np.shape(seismogram1)
75
76 for i in range(n):
77     y1[:,i] = konvin3190(h1,seismogram1[:,i],0)
78
79 fs = 1/(t[2]-t[1])
80 dt = 1/fs
81
82 hastighet = 2000
83
84 vnmo = np.full((1,601),hastighet)
85
86 seismno,r1,r2 = nmocorrection(t,dt,offset1,y1,vnmo)
87
88 plt.imshow(seismno,cmap='gray')
89 plt.colorbar()
90 plt.show()

```

Og får ved $v = \frac{1667.67m}{s}$:



Og får ved $v = \frac{2000m}{s}$:

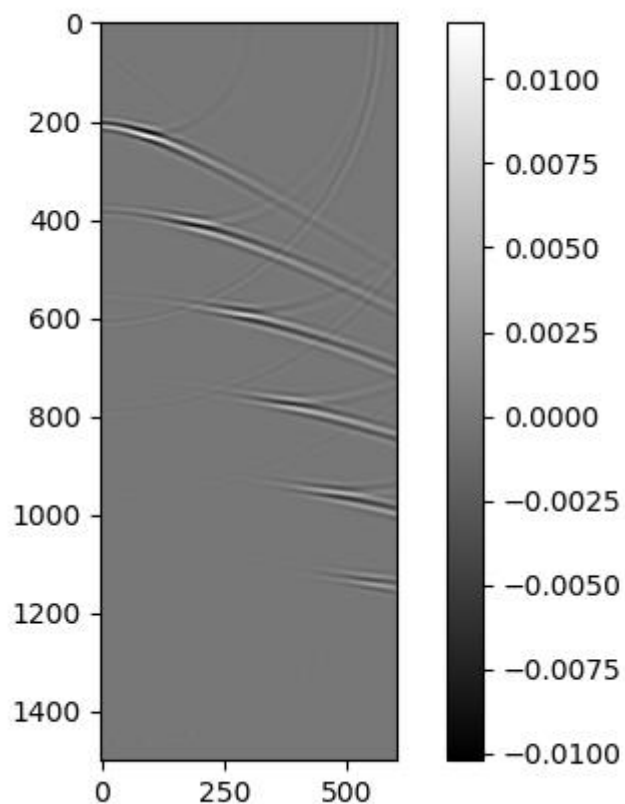


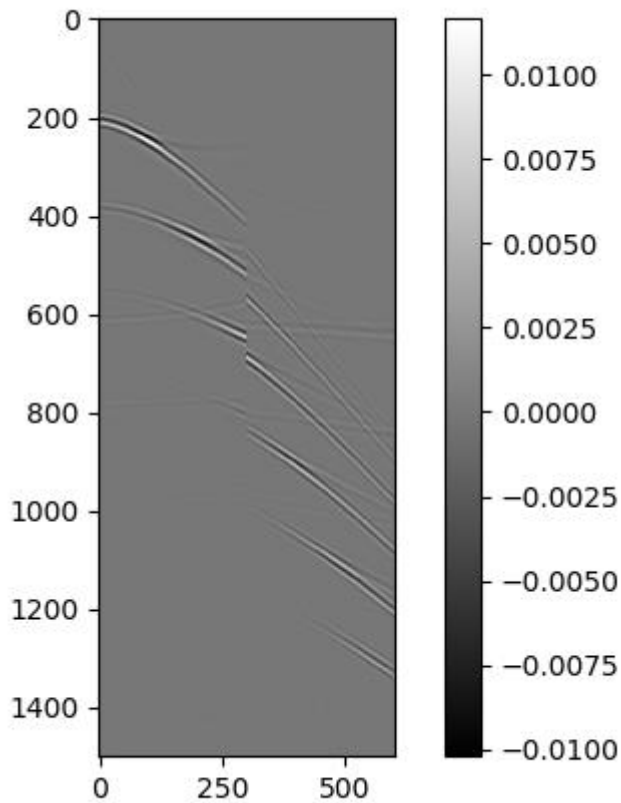
Vi ser at ved $v = \frac{2000m}{s}$ for vi et flatere havbunnsrefleksjon, og vi bruker det derfor i videre utregning.

- 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

```
1 vnmo2 = np.full((1,601),2500)
2 for i in range(300):
3     vnmo2[0,i] = 2000
4     seismno2,r12,r22 = nmocorrection(t,dt,offset1,y1,vnmo2)
5
6 plt.imshow(seismno2,cmap='gray')
7 plt.colorbar()
8 plt.show()
```

Kjøreeksempel gir:





Oppgave 7 (Sedimenthastighet II)

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

$$\begin{aligned} t_1 &= 900s \\ t_2 &= 2100s \\ x_1 &= 5100 \\ x_2 &= 4900 \end{aligned}$$

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:

$$\begin{aligned} t_1 &= 100s \\ t_2 &= 1800s \\ x_1 &= 4500 \\ x_2 &= 4100 \end{aligned}$$

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:

```
1 m,n = shape(seismogram2)
2
3 y2 = np.zeros_like(seismogram2)
4
```

```

5  for i in range(n):
6      y2[:,i] = konvin3190(seismogram2[:,i],h1,0)
7
8  plt.imshow(y2)
9  plt.colorbar()
10 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