Hjemmeopgave 3

34210 Digital Kommunikation

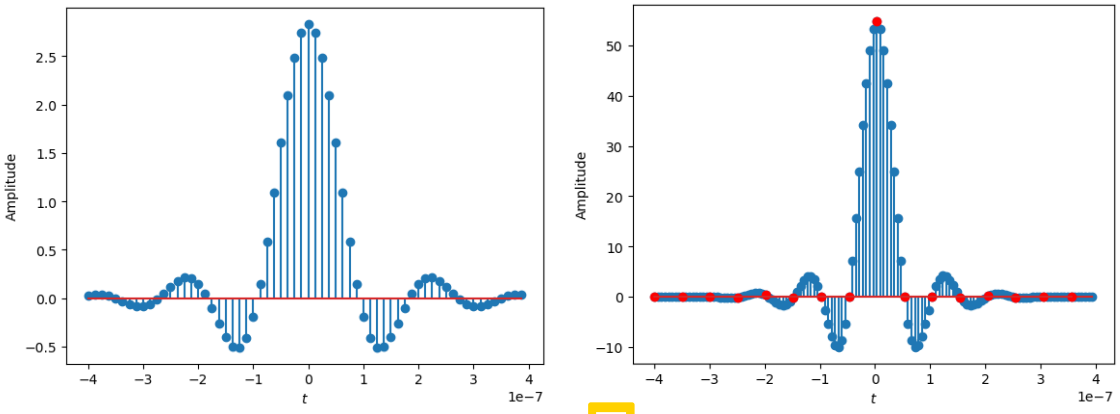
Daniel Brasholt s214676

April 2023

Indhold

Spørgsmål 1 Pulsformen

Spm 1.1: g og Synkpunkt



(a) Pulsform for $g(t)$ *(b) Pulsform for $x(t)$*

Figur 1: *g og x tegnet med `stem`-funktionen.*

På ovenstående figur 1b kan man se, at `x` har sin maksimale værdi i midten, hvilket svarer til sample nummer 65 - i bilagene markeret som 64 på grund af 0-indeksering. Derfor sættes:

$$\texttt{Synkpunkt} = 64$$
$$\texttt{E} \approx 54.8$$

Spm 1.2: Intersymbolinterferens

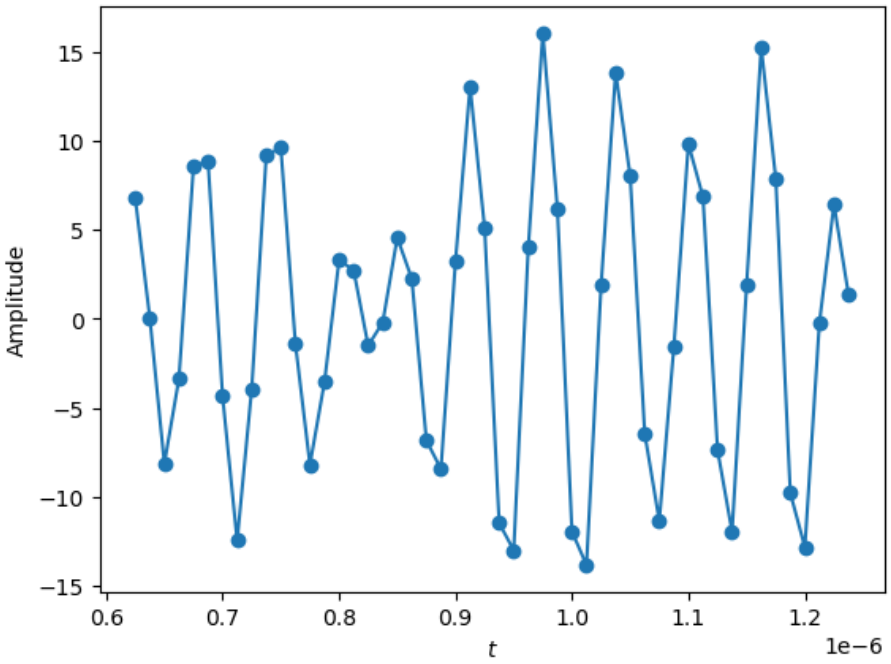
Den pulsform, der er valgt til dette signal, er kvadratroden af en cosinus roll-off funktion. Denne funktion har den fordel, at den for nT, $n \neq 0$, går gennem 0. Vælger vi da det rigtige tidspunkt at sample (`Synkpunkt`) vil vi kunne genskabe signalet fuldstændigt, såfremt der ikke er støj. Da vi sampler med mellemrum $Ts = T/m = T/8$, vil hver 8. sample gå gennem 0, hvilket er fremhævet på figur 1b med røde punkter.

Spørgsmål 2 Modulation med 16QAM

Spm 2.1: Frembringelse af v

Af bilagene fremgår det, hvordan det transmitterede signal er dannet. Dog er der ikke et plot med af dette, da det ikke ville vise noget interessant.

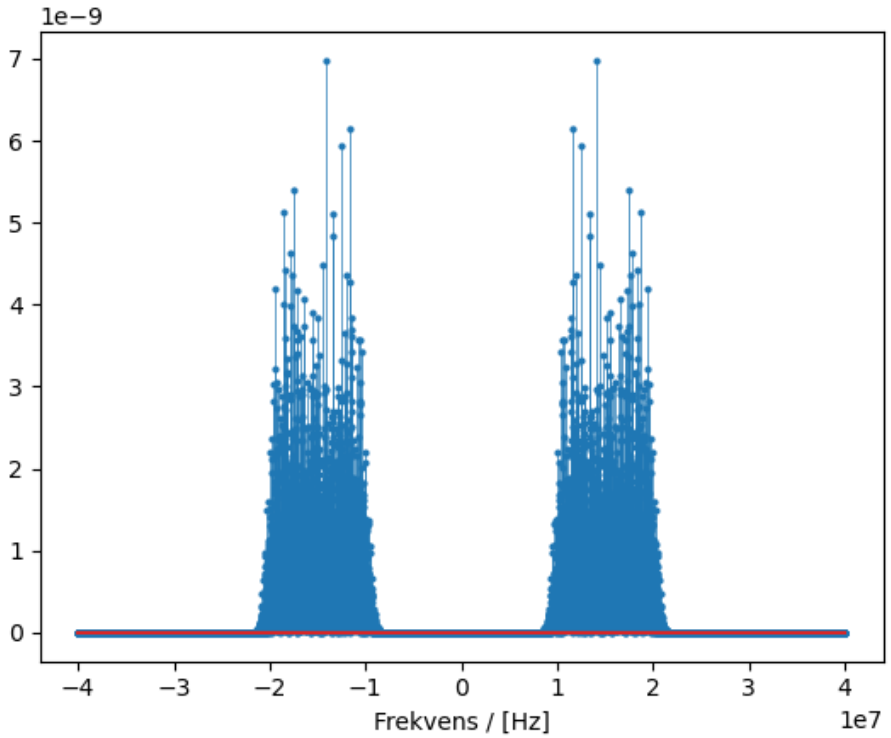
Spm 2.2: Fase- og amplitudeskift i v



Figur 2: *Udsnit af v fra indeks* 50 *til* 100

Ovenstående figur 2 viser et udsnit af v. Med afstand `T` bør man kunne se skift i fase og/eller amplitude, da det er her, signalet indeholder et symbol. For eksempel kan sådan et faseskift ses omkring $t = 0.85$. Herfra er der yderligere skift i amplitude og fase. Faseforskydningerne viser sig ved at samplepunkterne lægger sig anderledes på hver bølge, hvilket indikerer, at tiden er forskudt.

Spm 2.3: Energispektrum af v



Figur 3: *Energispektrum af v*

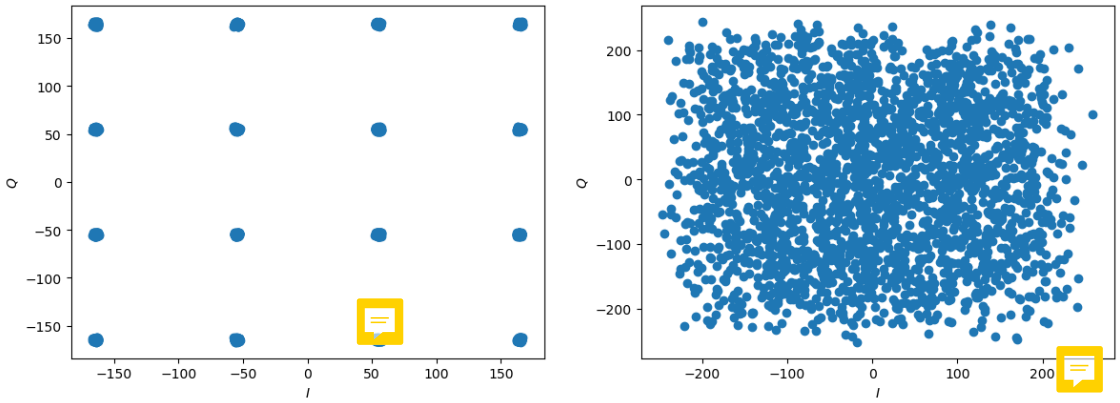
På figur 3 kan det ses, at spektret har to bidrag symmetriske om $0Hz$. Disse to bidrag har midtpunkt ved cirka $\pm 1.5 \cdot 10^7 Hz = \pm 15MHz$. Dette passer med, at vi må forvente, at der er bidrag omkring frekvensen på bærebølgen, $f_c = 15MHz$.

Bredden på hver af disse bidrag er cirka fra lidt under $10MHz$ til lidt over $20MHz$. Dette stemmer overens med, at vi har brugt sinus og cosinus på måden $\sqrt{2}\cos 2\pi f_c t$. Sinus og cosinus går hver mellem ± 1, og $\sqrt{2} \approx 1.4$, så vi ender med at modulere bærebølgen med cirka den faktor - det ender med at ligge mellem cirka $21.2MHz$ og $8.8MHz$, hvilket stemmer overens med spektret på figur 3.

Til sidst kan man lægge mærke til, at der er flest bidrag *omkring* $\pm 15MHz$ og ikke meget *ved* $15MHz$. Det kan forklares ved at vi modulerer omkring bærebølgen med faktoren beskrevet ovenfor og derfor ikke ofte lander netop ovenpå den. Dette giver de to „toppe“ man ser omkring f_c.

Spørgsmål 3 Modtageren

Spm 3.1: Betydningen af det rigtige Synkpunkt

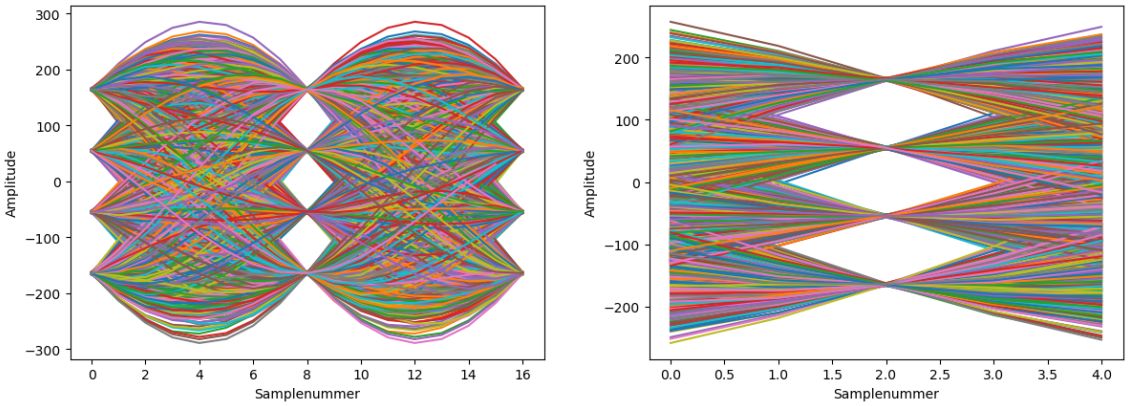


(a) I og Q med det rigtige Synkpunkt *(b) I og Q med Synkpunkt forskudt med 10*

Figur 4: *Betydningen af at vælge det rigtige Synkpunkt til at sample hos modtageren*

På ovenstående figur 4a kan det ses, at værdierne fra I og Q ganske rigtigt lægger sig som punkterne i Figur 6.10 fra noterne, dog med 16 punkter i stedet for 4, da vi bruger 16QAM i stedet for QPSK. Vælger vi det forkerte `Synkpunkt`, ligger punkterne med for meget støj til, at symbolerne kan tydes, som illustreret på figur 4b.

Spm 3.2: Øjediagram for modtageren



(a) Øjediagram over 16 samples *(b) Øjediagram over 4 samples*

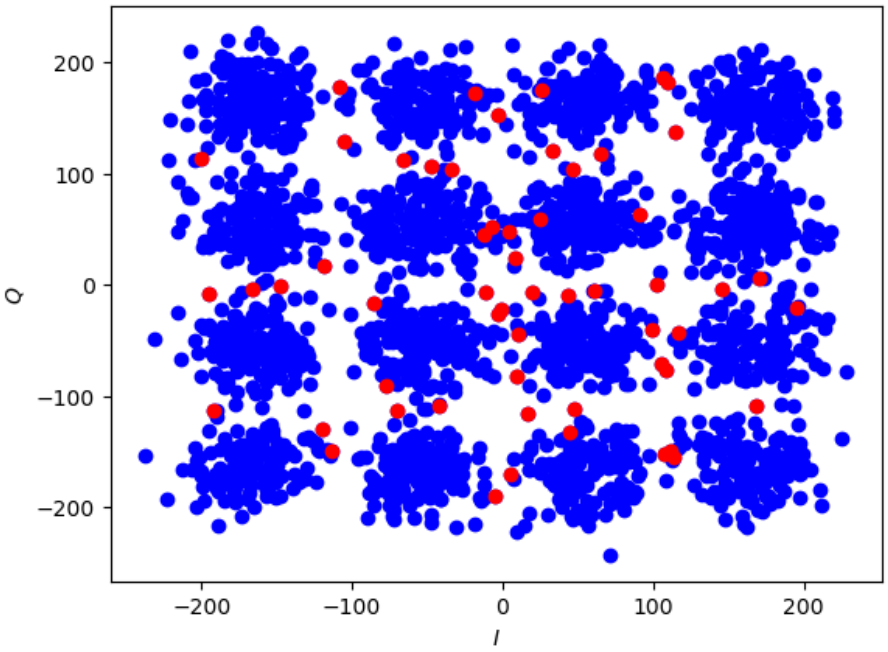
Figur 5: *Øjediagrammer over forskellige antal samples for bedre at vise øjets form*

På ovenstående øjediagram, figur 5, kan det ses, at signalet kun ligger „pænt“ netop i de punkter, hvori der skal samples hos modtageren. Dette underbygger også figur 4b, hvor punkterne ikke ligger pænt, hvis der vælges det forkerte `Synkpunkt`. Dog er signalet perfekt med mellemrum T, så modtageren kan sample symbolerne uden at fortolke symbolerne forkert.

Spm 3.3: Gendannelse af a og b

Da punkterne ligger så klart defineret, som vist på figur 4a, kan sekvenserne a og b let genskabes fra I og Q, såfremt `Synkpunkt` vælges rigtigt. Med Numpy er det talt, at der er 0 fejl, hvilket fremgår af bilagene.

Spørgsmål 4 Støj på signalet

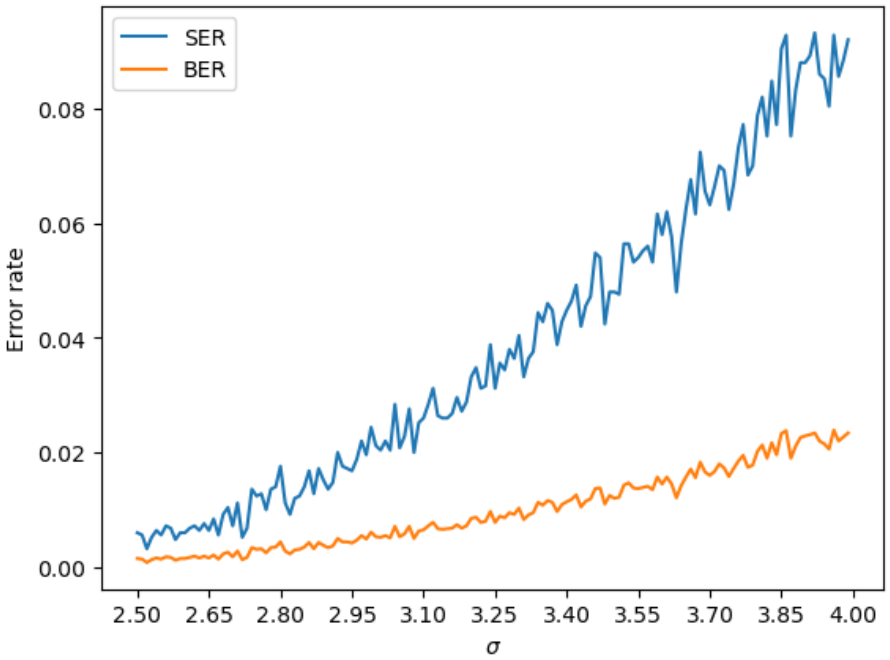


Figur 6: *Modtagne symboler efter addition med normalfordelt støj*

Figur 6 viser skyerne omkring konstellationspunkterne, der fremkommer, når der føjes støj til signalet. Derudover er der markeret de symboler, for hvilke der er truffet forkerte beslutninger. Der har i alt været 31 fejl på a og 26 fejl på b, hvilket i alt giver 57 fejl[1].

[1] Tallene i bilagene afviger muligvis fra disse, da koden ved hver gennemgang giver forskellige støjværdier. Plottene afviger af denne grund muligvis også.

Spørgsmål 5 Fejlhyppighed ved forskellige σ



Figur 7: *Bit- og symbolfejlhyppighed som funktion af støj, σ*

Det kan ses på figur 7, at bitfejlsandsynligheden ganske rigtigt er omkring $\frac{1}{4}$ af symbolfejlhyppigheden. Dette er eftersom symbolerne er Gray-kodede, hvilket betyder, at de symboler, der ligger lige ved siden af hinanden, kun har 1 bit til forskel. Da er der meget lille risiko for, at en beslutning bliver så forkert, at man får 2 bitfejl for 1 symbolfejl. Ser man på figur 4 vil det kræve, at et symbol kravler „diagonalt“, for at det giver 2 bitfejl. Dette er sket, hvilket er vist i bilagene, men det er stadig sjældent. Eftersom antallet af symbolfejl og bitfejl er omtrent det samme, vil bitfejlhyppigheden være cirka $\frac{1}{4}$ af symbolfejlhyppigheden, da vi i førstnævnte dividerer med L (antallet af bit) og i sidstnævnte med $L/4$ (antallet af symboler). Som forventet bliver fejl også mere hyppige ved højere σ.

Hjemmeopgave 3

April 24, 2023

1 Imports

[39]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

2 Spørgsmål 1

[2]:
```
T = 1/10_000_000
a = 0.3
m = 8
Ts = T/m
sampling_f = 1/Ts
```

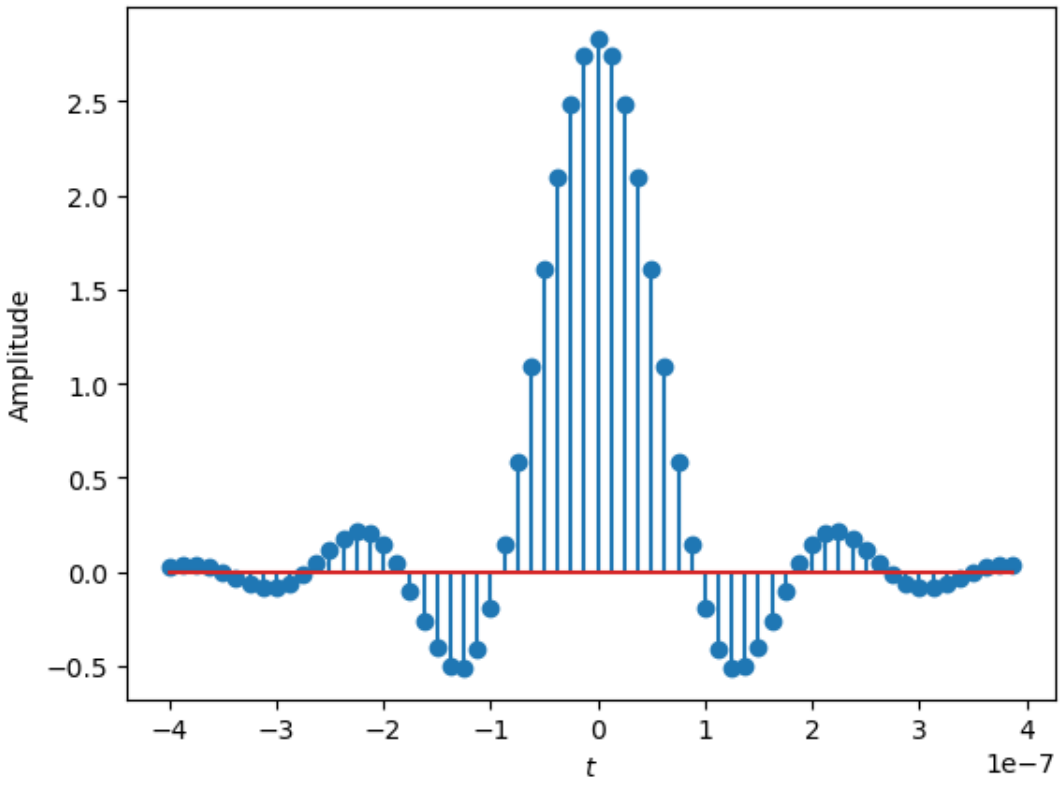
[3]:
```
ti = np.arange(-4*T, 4*T, Ts)
```

$$g(t) = \frac{cos\left(\frac{(1+a)\pi ti}{T}\right) + \frac{\pi(1-a)}{4a}\,\mathrm{sinc}\left(\frac{(1-a)ti}{T}\right)}{1-\left(\frac{4ati}{T}\right)^2}$$

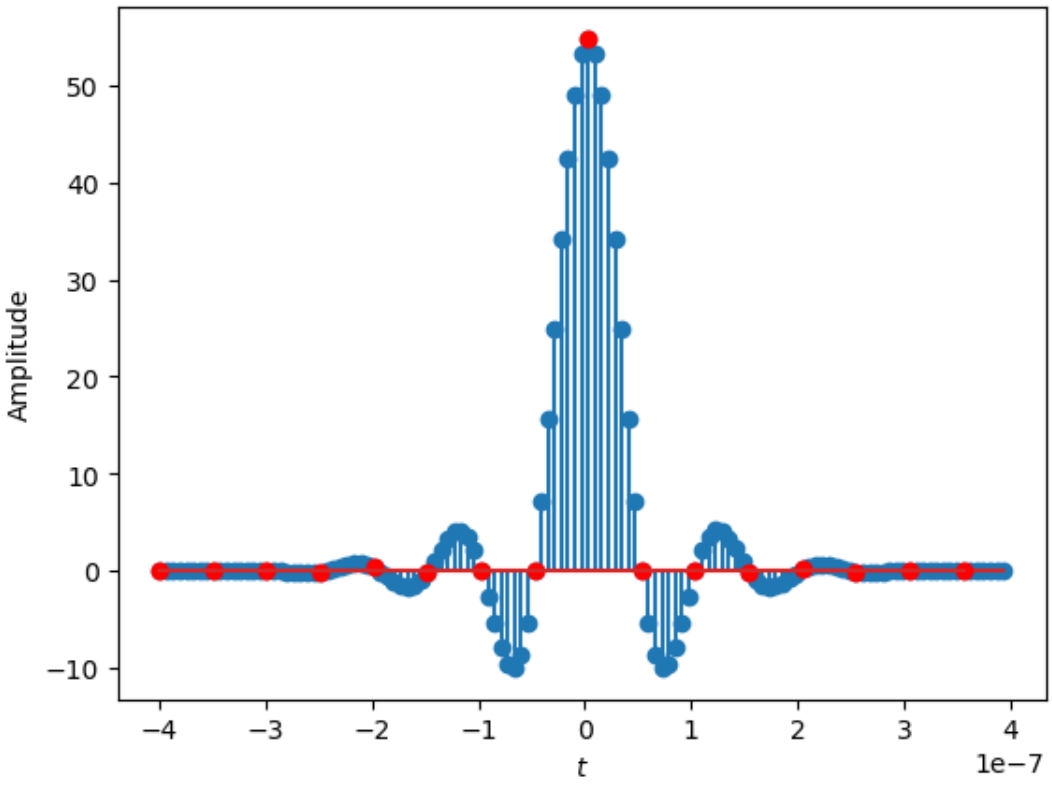
[4]:
```
g = ((np.cos((1+a)*np.pi*ti/T)
       + (np.pi*(1-a)/(4*a))
       * np.sinc((1-a)*ti/T))
        / (1-(4*a*ti/T)**2))
```

[5]:
```
x = np.convolve(g,g)
```

[31]:
```
x_axis = np.arange(-4*T,4*T,Ts)
plt.stem(x_axis, g)
plt.xlabel('$t$')
plt.ylabel('Amplitude')
plt.show()
```



[36]:
```
x_axis = np.arange(-4*T, 4*T, 8*T/127)
plt.stem(x_axis, x)
plt.stem(x_axis[::8], x[::8], markerfmt='r')
plt.xlabel('$t$')
plt.ylabel('Amplitude')
plt.show()
```



[8]:
```
Synkpunkt, E = np.argmax(x), np.max(x)
print(f'{Synkpunkt=}, {E=}')
```

```
Synkpunkt=64, E=54.81142578650048
```

3 Spørgsmål 2

[9]:
```
L = 10000
c = np.random.choice([0, 1], (L//4,4), replace=True)
```

[10]:
```
a = [-3 if all(i==[0,0])
    else -1 if all(i==[0,1])
    else 1 if all(i==[1,1])
    else 3 for i in c[:,2:]]
b = [-3 if all(i==[0,0])
    else -1 if all(i==[0,1])
    else 1 if all(i==[1,1])
    else 3 for i in c[:,:2]]
a,b = np.array(a),np.array(b)
```

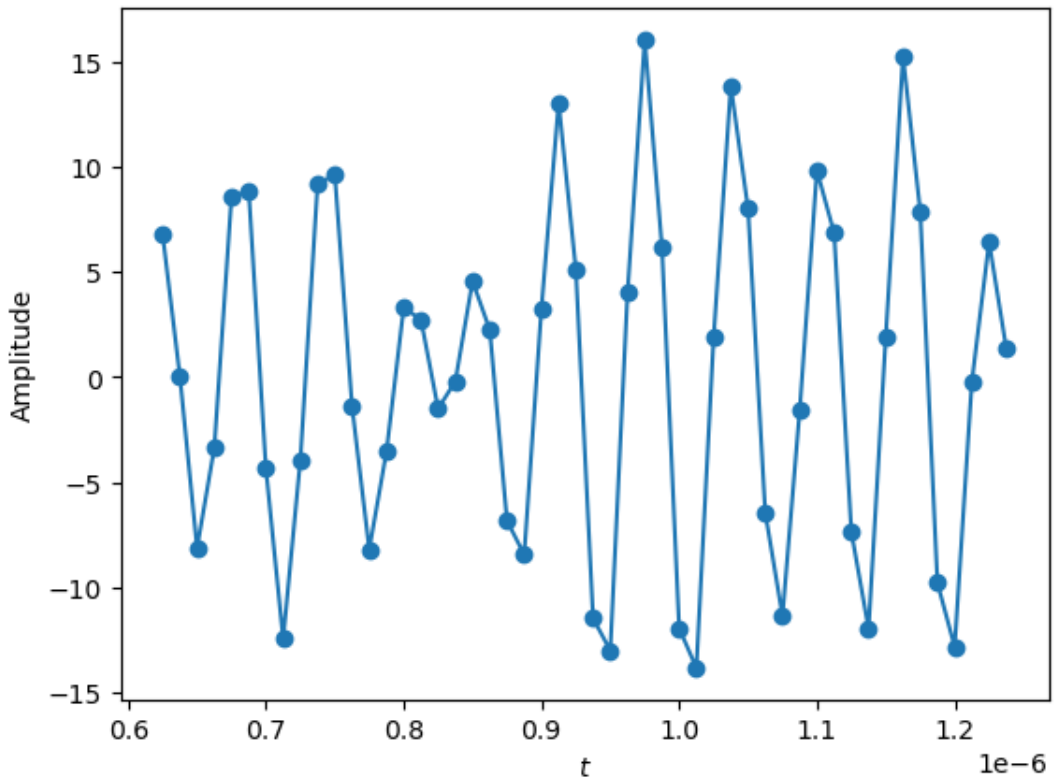
[11]:
```
a0 = np.zeros(a.size*m - (m-1), a.dtype)
a0[::m] = a
va = np.convolve(g,a0)
b0 = np.zeros(b.size*m - (m-1), b.dtype)
b0[::m] = b
vb = np.convolve(g,b0)
```

[12]:
```
N = va.size
t = np.arange(0, N*Ts, Ts)
fc = 15_000_000
```

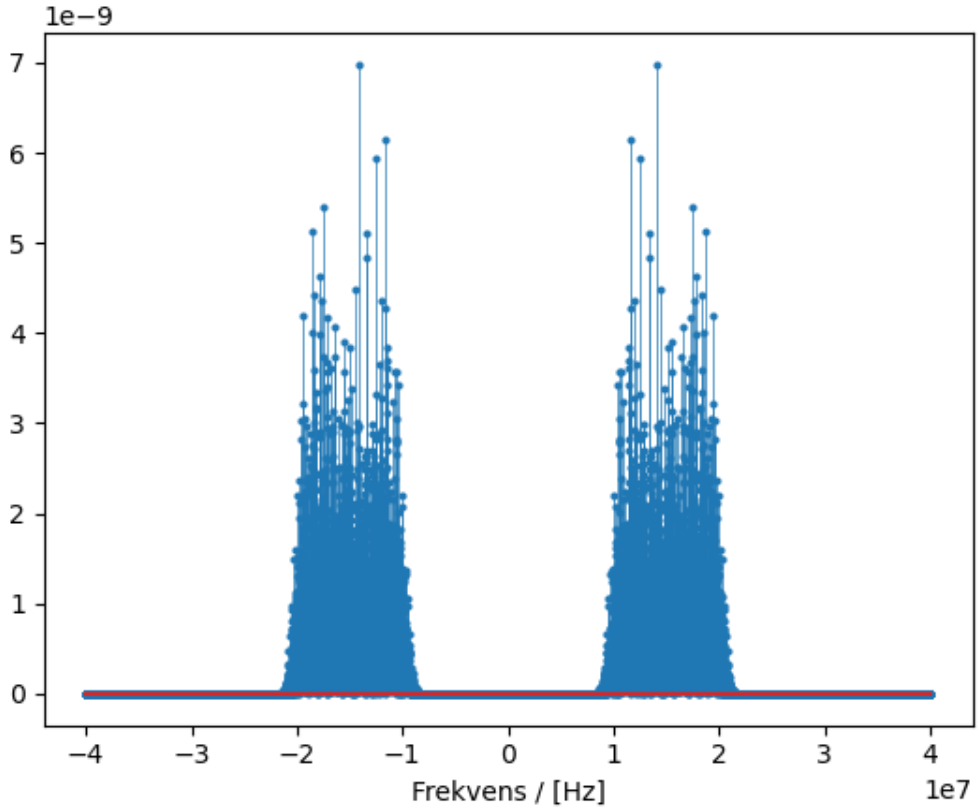
[13]:
```
va_modulated = va * np.sqrt(2)*np.cos(2*np.pi*fc*t)
vb_modulated = vb * np.sqrt(2)*np.sin(2*np.pi*fc*t)
```

[14]:
```
v = va_modulated + vb_modulated
```

[45]:
```
start, end = 50, 100
plt.plot(t[start:end], v[start:end])
plt.scatter(t[start:end], v[start:end])
plt.xlabel('$t$')
plt.ylabel('Amplitude')
plt.show()
```



[49]:
```
v_fft = np.fft.fft(v) * Ts
v_energy = np.abs(v_fft)**2
f_axis = np.arange(-N/2, N/2) / (N*Ts)
markers,stems,base = plt.stem(f_axis, np.fft.fftshift(v_energy))
plt.setp(stems, 'linewidth', 0.5)
plt.setp(markers, markersize=2)
plt.xlabel('Frekvens / [Hz]')
plt.show()
```



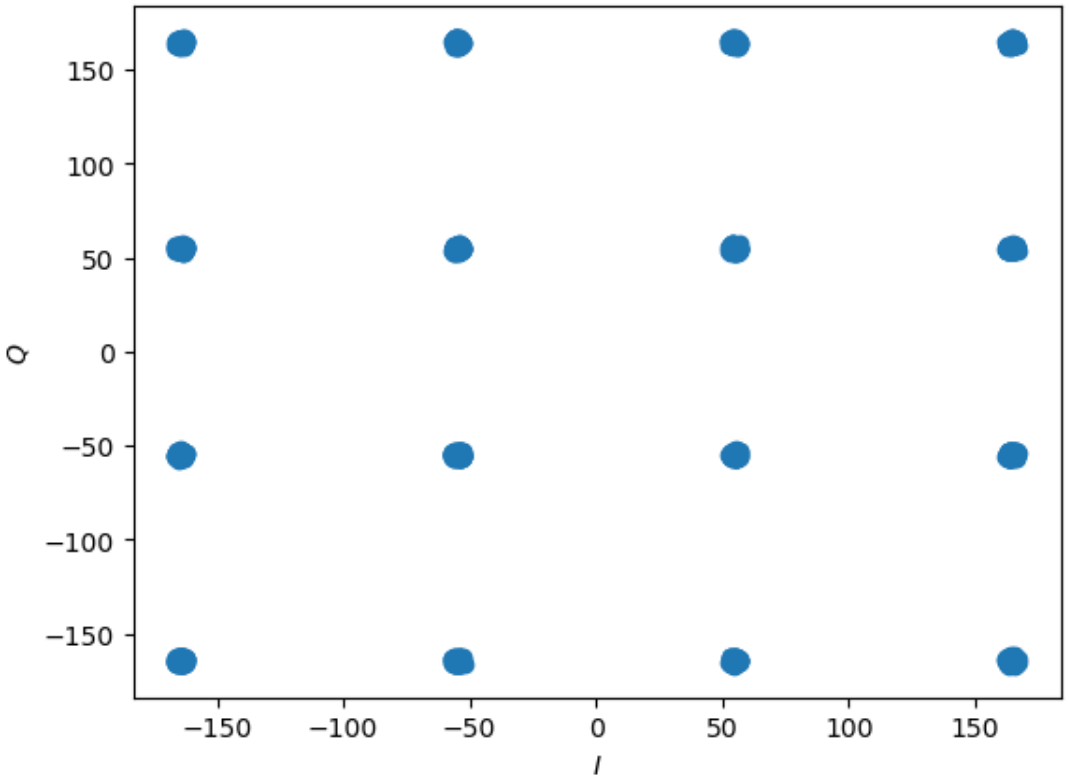
4 Spørgsmål 3

[16]:
```
i = v*np.sqrt(2)*np.cos(2*np.pi*fc*t)
q = v*np.sqrt(2)*np.sin(2*np.pi*fc*t)
I = np.convolve(g, i)
Q = np.convolve(g, q)
```

[17]:
```
I.size - (2500*m+Synkpunkt)
```

[17]: 55

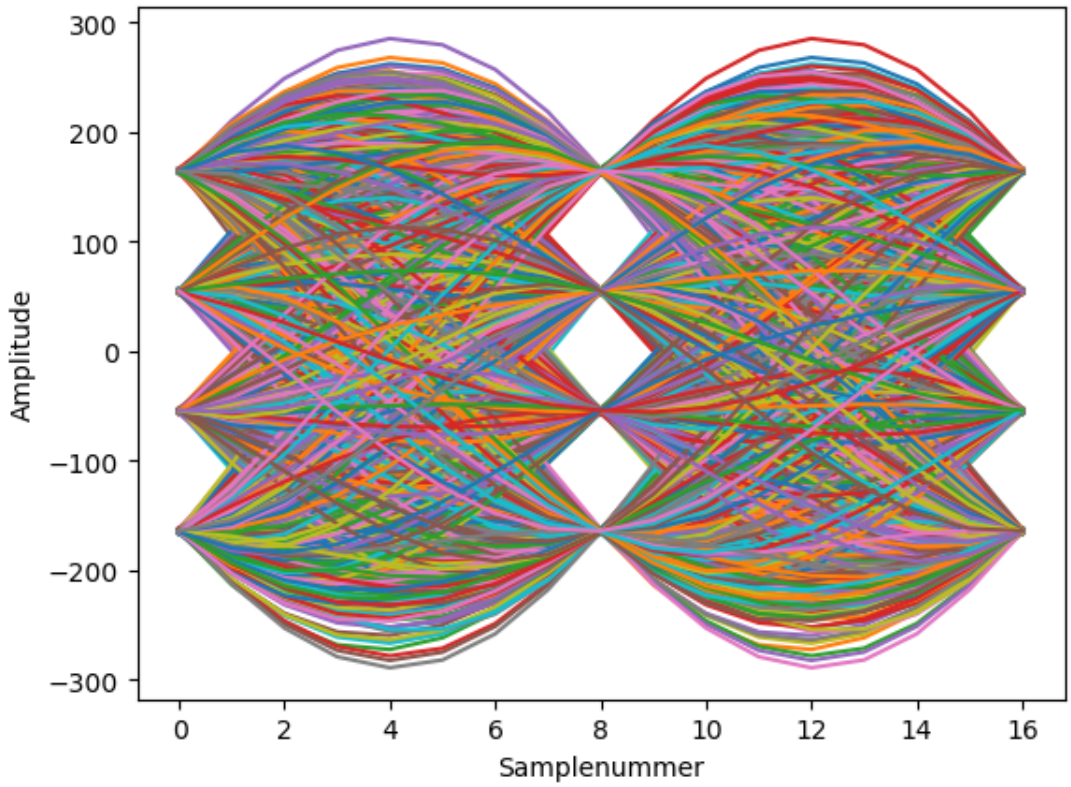
[18]:
```
temp_synk = Synkpunkt # her -10 for det forkerte synkpunkt
stop_idx = I.size - (2500*m+temp_synk)
I_sliced = I[temp_synk:-stop_idx:m]
Q_sliced = Q[temp_synk:-stop_idx:m]
plt.scatter(I_sliced,Q_sliced)
plt.xlabel('$I$')
plt.ylabel('$Q$')
plt.show()
```



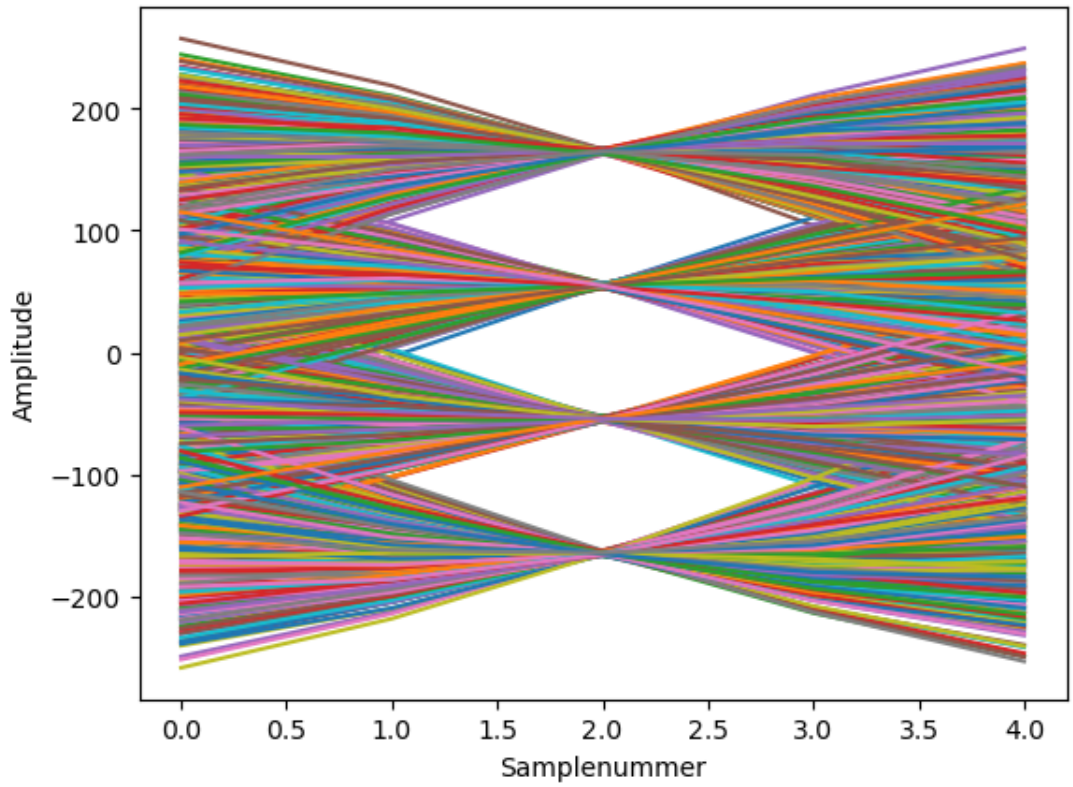
[19]:
```
I_sliced.shape
```

[19]: (2500,)

[27]:
```
I_eye = I[Synkpunkt+m-4:-stop_idx-m]
for i in range(I_eye.size//m):
    plt.plot(I_eye[i*m+4:(i+3)*m-3])
plt.xlabel('Samplenummer')
plt.ylabel('Amplitude')
plt.show()
```



[28]:
```
I_eye = I[Synkpunkt+m-4:-stop_idx-m]
for i in range(I_eye.size//m):
    plt.plot(I_eye[i*m+2:(i+1)*m-1])
plt.xlabel('Samplenummer')
plt.ylabel('Amplitude')
plt.show()
```



[19]:
```
I_sliced.size
```

[19]: 2500

[20]:
```
a_k = [-3 if i<-2*E
      else -1 if -2*E<i<0
      else 1 if 0<i<2*E
      else 3 for i in I_sliced]
b_k = [-3 if i<-2*E
      else -1 if -2*E<i<0
      else 1 if 0<i<2*E
      else 3 for i in Q_sliced]
```

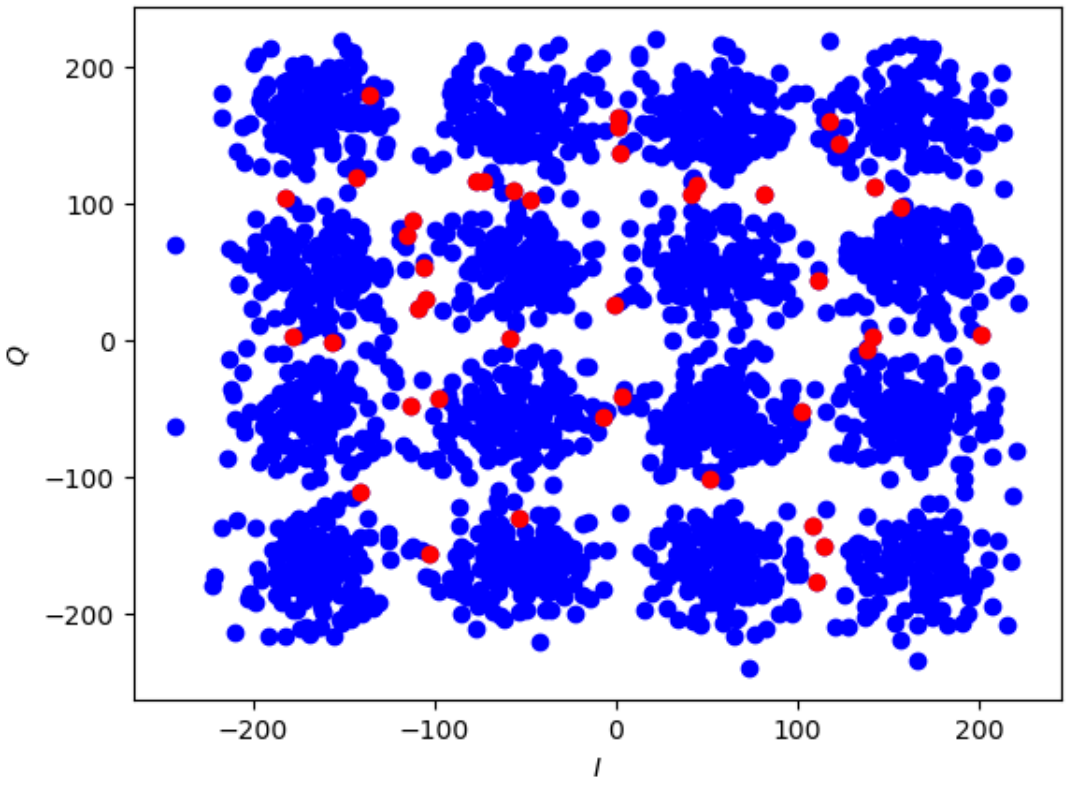
[21]:
```
print(f'Fejl på a og I: {sum(a != a_k)}')
print(f'Fejl på b og Q: {sum(b != b_k)}')
```

```
Fejl på a og I: 0
Fejl på b og Q: 0
```

5 Spørgsmål 4

[23]:
```
sigma = 3
noise = np.random.normal(loc=0, scale=sigma, size=v.size)
v_noisy = v + noise
i_noisy = v_noisy*np.sqrt(2)*np.cos(2*np.pi*fc*t)
q_noisy = v_noisy*np.sqrt(2)*np.sin(2*np.pi*fc*t)
I_noisy = np.convolve(g, i_noisy)
Q_noisy = np.convolve(g, q_noisy)
stop_idx = I_noisy.size - (2500*m+Synkpunkt)
I_noisy_sliced = I_noisy[Synkpunkt:-stop_idx:m]
Q_noisy_sliced = Q_noisy[Synkpunkt:-stop_idx:m]
a_k_noisy = [-3 if i<-2*E
      else -1 if -2*E<i<0
      else 1 if 0<i<2*E
      else 3 for i in I_noisy_sliced]
b_k_noisy = [-3 if i<-2*E
      else -1 if -2*E<i<0
      else 1 if 0<i<2*E
      else 3 for i in Q_noisy_sliced]
print(f'Fejl på a og I: {sum(a != a_k_noisy)}')
print(f'Fejl på b og Q: {sum(b != b_k_noisy)}')
wrong_on_a = np.nonzero(a!=a_k_noisy)
wrong_on_b = np.nonzero(b!=b_k_noisy)
plt.scatter(I_noisy_sliced,Q_noisy_sliced, c='b')
plt.scatter(I_noisy_sliced[wrong_on_a], Q_noisy_sliced[wrong_on_a], c='r')
plt.scatter(I_noisy_sliced[wrong_on_b], Q_noisy_sliced[wrong_on_b], c='r')
plt.xlabel('$I$')
plt.ylabel('$Q$')
plt.show()
```

```
Fejl på a og I: 22
Fejl på b og Q: 20
```



[25]:
```
a_b_pairs = [(i,j) for i,j in zip(a,b)]
a_b_k_pairs = [(i,j) for i,j in zip(a_k_noisy,b_k_noisy)]
```

[26]:
```
errors = sum([i!=j for i,j in zip(a_b_pairs,a_b_k_pairs)])
print(f'A total of {errors} errors')
```

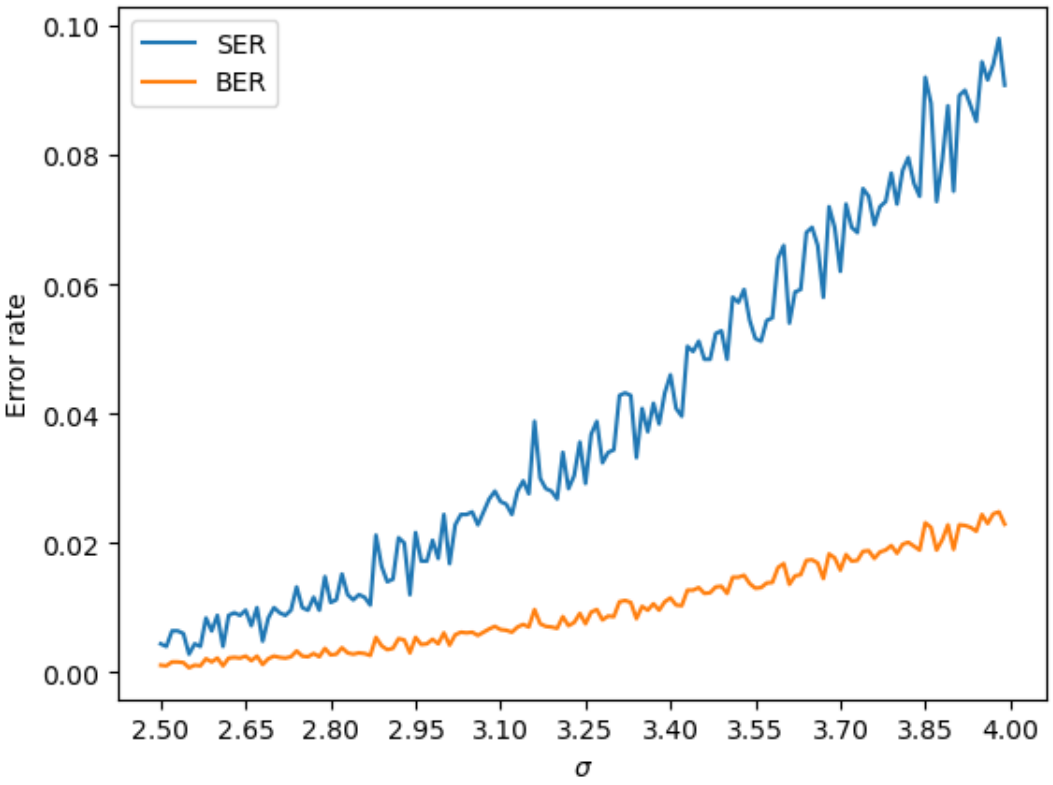
```
A total of 42 errors
```

6 Spørgsmål 5

[27]:
```
translation = {
    -3: [0,0],
    -1: [0,1],
    1: [1,1],
    3: [1,0]
}
symbol_error_list = []
bit_error_list = []
sigmas = np.arange(2.5,4,0.01)
for sigma in sigmas:
    noise = np.random.normal(loc=0, scale=sigma, size=v.size)
```

```
    v_noisy = v + noise
    i_noisy = v_noisy*np.sqrt(2)*np.cos(2*np.pi*fc*t)
    q_noisy = v_noisy*np.sqrt(2)*np.sin(2*np.pi*fc*t)
    I_noisy = np.convolve(g, i_noisy)
    Q_noisy = np.convolve(g, q_noisy)
    stop_idx = I_noisy.size - (2500*m+Synkpunkt)
    I_noisy_sliced = I_noisy[Synkpunkt:-stop_idx:m]
    Q_noisy_sliced = Q_noisy[Synkpunkt:-stop_idx:m]
    a_k_noisy = [-3 if i<-2*E
          else -1 if -2*E<i<0
          else 1 if 0<i<2*E
          else 3 for i in I_noisy_sliced]
    b_k_noisy = [-3 if i<-2*E
          else -1 if -2*E<i<0
          else 1 if 0<i<2*E
          else 3 for i in Q_noisy_sliced]
    a_b_k_pairs = [(i,j) for i,j in zip(a_k_noisy,b_k_noisy)]
    symbol_errors = sum([i!=j for i,j in zip(a_b_pairs,a_b_k_pairs)])
    symbol_error_list.append(symbol_errors)
    a_translated, b_translated = np.array([translation[i] for i in a_k_noisy]),␣
 ↪\
                                 np.array([translation[i] for i in b_k_noisy])
    received_bits_matrix = np.column_stack((b_translated, a_translated))
    bit_errors = sum(sum(received_bits_matrix != c))
    bit_error_list.append(bit_errors)
symbol_error_list = np.array(symbol_error_list)
bit_error_list = np.array(bit_error_list)
```

[28]:
```
plt.plot(sigmas, symbol_error_list/(L/4), label='SER')
plt.plot(sigmas, bit_error_list/L, label='BER')
plt.xlabel('$\sigma$')
plt.ylabel('Error rate')
plt.xticks(np.arange(2.5,4.01,0.15))
plt.legend()
plt.show()
```


SER
BER
Error rate
0.10
0.08
0.06
0.04
0.02
0.00
2.50
2.65
2.80
2.95
3.10
3.25
3.40
3.55
3.70
3.85
4.00
σ