

DSA Case 1- FSK

Navn	Studienummer
Naiyun Wu	201405716
Frederik Gadegaard	201405625
Maciej Lech	20107672
Jacob Aagaard	201404442

Indhold

Opgave 1 – Signal generation / kodning.....	3
Genererer lydsignal med FSKgenerator	3
Analyser signalet i tids- og frekvens-domænet.	3
Analyser signalet vha. Short-Time Fourier Transform.....	5
FSKgenerator input-parametrene.	7
Opgave 2 – Dekodning.....	8
Dekodning af signal	8
Vores egen DFT.....	8
Opgave 3 – Signal-støj-forhold	10
Opgave 4 – Bit rate	12
Ændre på T_{symbol}	12
Hvad er betydningen af vindueslængden for amplitude-spektret af en sinus-tone?	13
Sende med forskellige frekvenser samtidig	13

Opgave 1 – Signal generation / kodning

Genererer lydsignal med FSKgenerator

Vi anvender funktionen FSKgenerator som vist i kodeudsnittet i Figur 1. Det ses at vi sender sekvensen 'easy', som vi igennem resten af denne case benytter os af. Vi har her et frekvensbånd fra 500 Hz til 3000Hz og en længde af signalet på $0.5 \cdot 4 = 2$ sekunder.

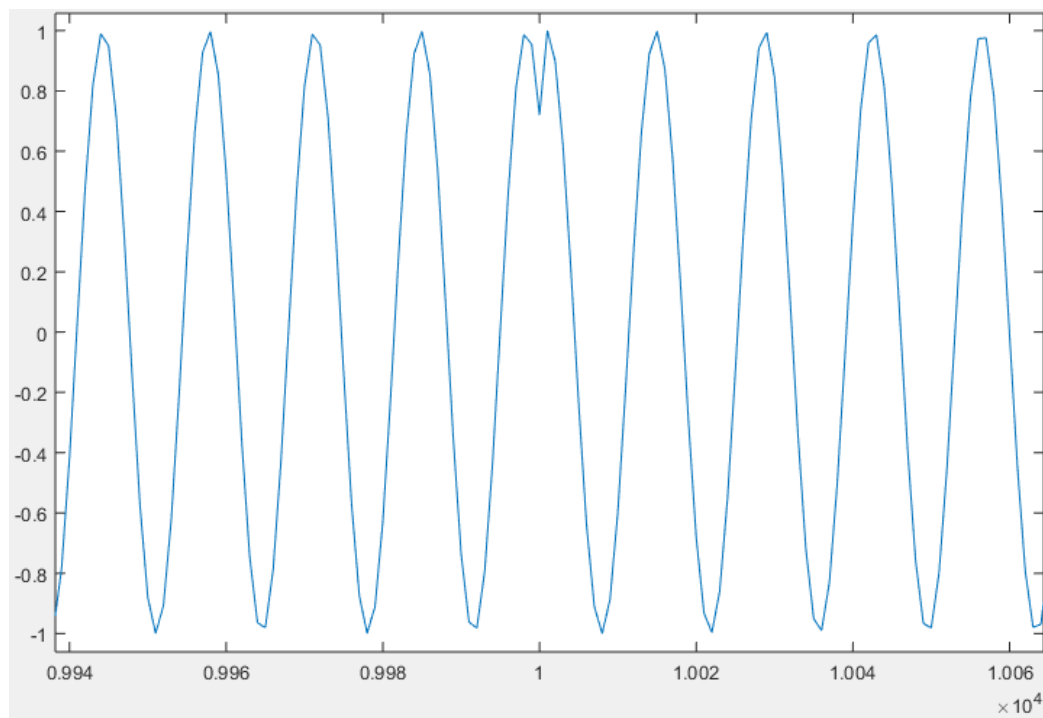
```
fstart = 500; % transmission band frequency start
fend = 3000; % transmission band frequency end
Tsymbol = 0.5; % symbol duration in seconds
fs = 20000; % sampling frequency

x = FSKgenerator('easy', fstart, fend, Tsymbol, fs);
```

Figur 1 parametre for FSKgenerator funktionen

Analyse af signalet i tids- og frekvens-domænet.

Herunder ses hvordan frekvensen skifter fra bogstav 'e' til bogstav 'a' ved sample $N = 1 \cdot 10^4$ i tidsdomænet.



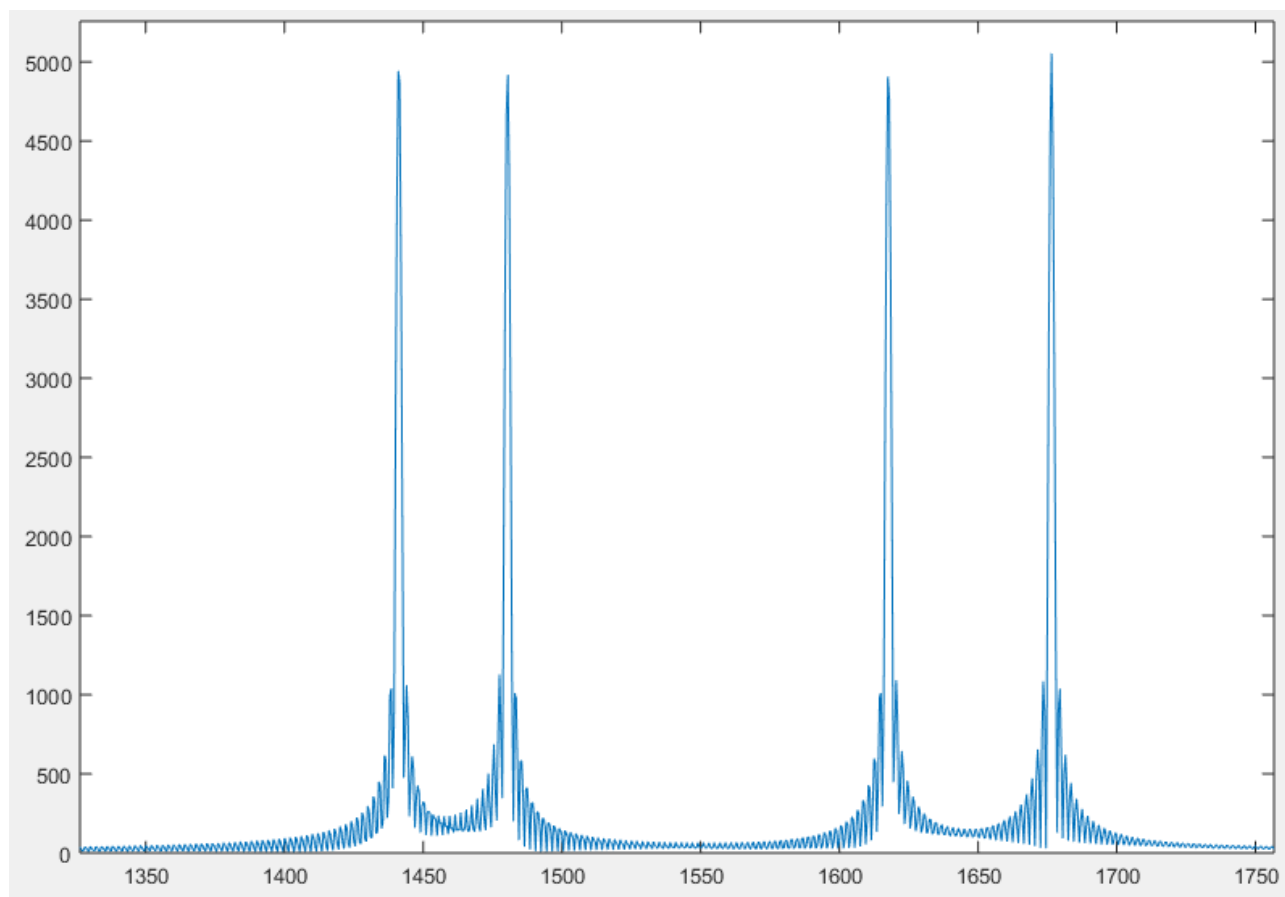
Figur 2 Billede af signal x i tidsdomænet

Da forskellen i decimal værdien af de to bogstaver er forholdsvis lille og med en frekvensopløsning på:

$$\text{frekvensopløsning} = \frac{f_{\text{end}} - f_{\text{start}}}{2^8} = 9.766$$

er det svært at se en forskel i frekvens i tidsdomænet.

Derfor er det nemmere at kigge på det i frekvensdomænet som vist i Figur 3.



Figur 3 signal x i frekvensdomænet

Her ses frekvensindholdet af signalet, og det er tydeligt at se de 4 forskellige bogstavers forskellige frekvenser. Disse er beregnet herunder i Figur 4, og sammenlignes de med Figur 3, ses det at det stemmer overens.

```

fstart:=500      res:=28=256
fslut:=3000
a:=97  e:=101   s:=115   y:=121

deltaF:=fslut-fstart

f(x):=x· $\frac{\text{deltaF}}{\text{res}}$ +fstart

f(e)=1486
f(a)=1447
f(s)=1623
f(y)=1682

```

Figur 4 Beregning af bogstavernes frekvenser

Vi har anvendt bogstaverne 'e', 'a', 's' og 'y', som har ascii-værdierne, 97, 101, 115 og 121 som vist i Figur 4. Disse ascii-værdier får tildelt en frekvens liggende indenfor frekvensbåndet angivet ved deltaF. Dette frekvens bånd deles op i 256 værdier, hvilket giver frekvensopløsningen.

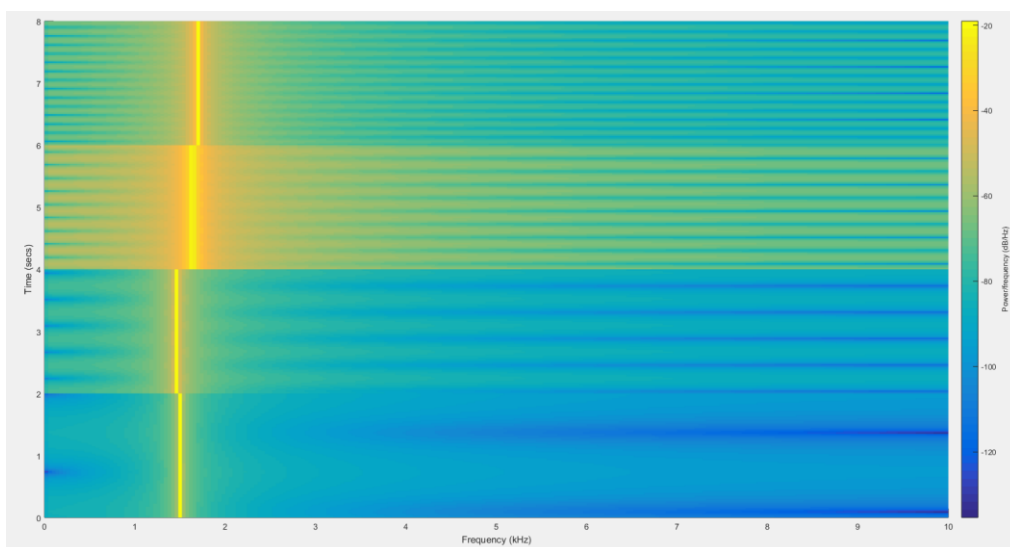
Analyser signalet vha. Short-Time Fourier Transform

```

fstart = 500; % transmission band frequency start
fend = 3000; % transmission band frequency end
Tsymbol = 2; % symbol duration in seconds
fs = 20000; % sampling frequency

```

Figur 5 parametre for FSKgenerator



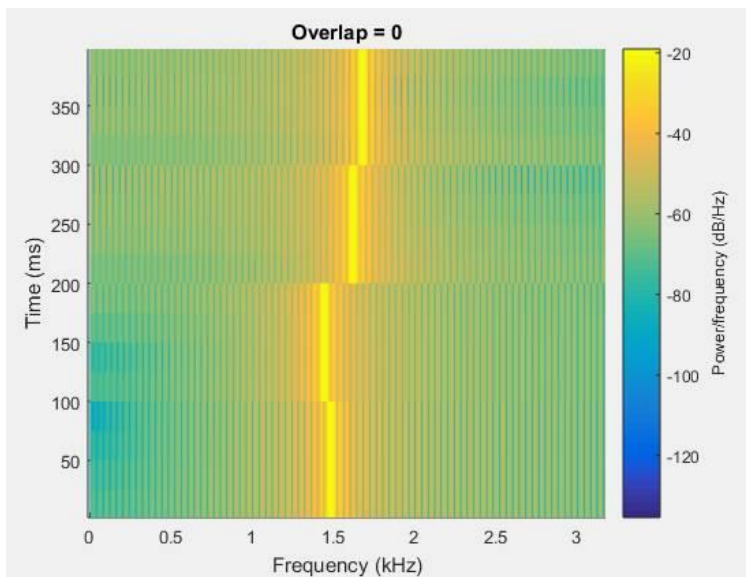
Figur 6 Spectrogram for signalet

Prototypen for spektrogrammet ses herunder:

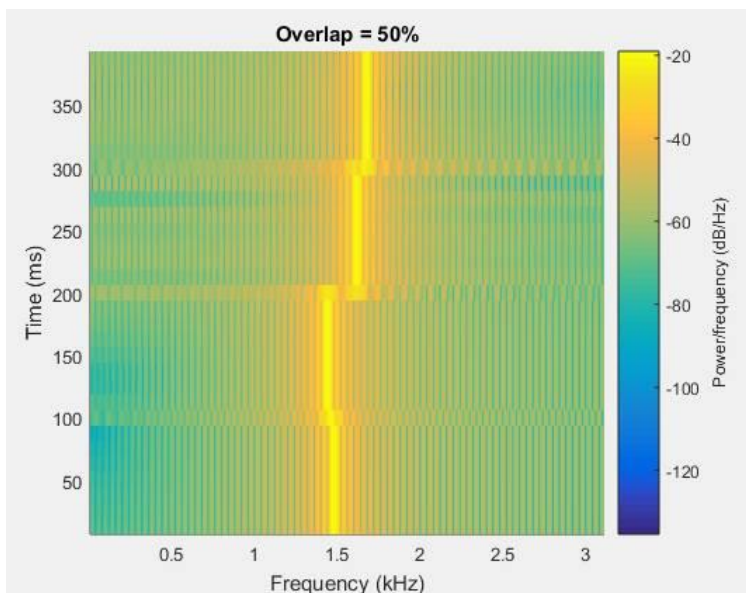
spectrogram(x, window, noverlap, nfft, fs);

'x' er signalet. 'Window' er den vinduesfunktion der ganges på signalet. 'fs' er samplefrekvensen og 'nfft' er antal DFT punkter, hvilket bestemmer frekvensopløsningen.

'noverlap' beskriver hvor mange samples som overlapper hinanden givet ud fra vindueslængden. Vi testede størrelse 'noverlap' og fandt ud af at ved den samme frekvensopløsning stiger tidsopløsningen proportionalt med den procentvise overlap af samples. Dette testede vi med en cursor, hvor vi ændrede x- og y-aksen (frekvens og tid), hvor tidsopløsningen blev 50% bedre ved 50% overlap. Dog virker det rent visuelt ikke som nogen fordel at anvende denne parametre, hvorfor vi blot undlod denne parameter.



Figur 7 noverlap = 0% af vindueslængden



Figur 8 noverlap = 50 % af vindueslængden

Tradeoff imellem opløsningen i tid og frekvensen kan vi bedst beskrive med et signal som bliver zeropadded.

Når vi zeropadder et signal øger vi antal samples N , og dermed falder værdien af frekvensopløsningen, da den er givet ved:

$$\Delta f = \frac{f_s}{N}$$

Dermed mindskes afstanden mellem hver enkelt bin i frekvensdomænet. Trade-off'et består i at vi har forlænget signalet med de zeropaddede værdier, og dermed stiger værdien af opløsningen i tid. Det betragtes i dette eksempel som en dårlig ting.

Ved en dårlig nok frekvensopløsning vil man f.eks. på figur 6 ikke kunne se hvornår frekvenserne ændrer sig, og dermed skelne bogstaverne fra hinanden. Hvis tidsopløsningen til gengæld ikke var god nok, vil man have svært ved at adskille tone-segmenterne fra hinanden.

Beskrivelse af FSKgenerator input-parametrene.

Fs:

Dette er samplefrekvensen som beskriver hvor hurtigt der samples. Det er vigtigt at samplefrekvensen er dobbelt så stor som f_{end} . For hvis en værdi af symbolet i 'mysymbolseq' er lig 256 tildeles dette symbol den maximale frekvens som er lig ' f_{end} ', og hvis denne skal samples korrekt skal $f_s = 2 * f_{\text{end}}$.

Fstart og fend:

Parametrene f_{start} og f_{end} definere i hvilket frekvensbånd der benyttes til at sende dataen henover.

Denne FSKgenerator er begrænset til at kunne pakke arrays med 256 forskellige symboler. Dette sker i linje 15 i FSKgenerator.m:

```
farray = linspace(fstart, fend, 256);
```

Dette begrænset indhold af forskellige symboler kan forøges ved at erstatte 256 med et større tal, og hvis der samtidig ønskes en større frekvensopløsning skal $\Delta f = f_{\text{end}} - f_{\text{start}}$ også være større.

Tsymbol:

Beskriver hvor længe frekvensen for hvert symbol sendes.

Opgave 2 – Dekodning

Denne del består af to dele, hhv. dekodningen signalet og DFT-algoritmen.

Dekodning af signal

```
1
2
3 function string = FSKdecode(fstart, fend, Tsymbol, fs, x)
4
5     string = []; freqArray=[];
6
7     Nsymbol = round((length(x)/fs)/Tsymbol); % calculate number of symbols received
8
9
10 for count=0:Nsymbol-1,
11
12     xCurrent = x(round((count*Tsymbol*fs+1):(count*Tsymbol*fs)+Tsymbol*fs)); % determines which part of signal x we analyze
13
14     y = Dekodning(xCurrent,fstart,fend,fs); % Our DFT function
15
16     y = abs(y);
17
18     y = y(1:length(y)/2);
19
20     max_y = max(y); % finding a peak-value where a frequency is located
21
22     freq = find(y == max_y,1); % finding the corresponding x-value which is the frequency-value
23
24     freq = freq/Tsymbol;
25
26     ascii = ceil(((freq-fstart)/((fend-fstart)/256))); % converts from frequency to ascii
27
28     string = [string char(ascii)];
29 end
30 return
```

Figur 9 matlab kode af dekodning funktionen

Dekodningen kan deles op i følgende faser: adskillelsen af signalet i tid af hvert bogstav, finde ud af hvilken frekvens signal-stykket indeholder og omregning tilbage til ascii – værdier/char-værdier.

”xCurrent” indeholder kun et signal svarende til ét bogstav af gangen. Dette signal analyseres i vores egen version af DFT i funktionen ”Dekodning” i linje 14. Herefter findes den største peak-værdi i frekvensindholdet af signal-stykket med funktionen ”max(x)”. Dette sætter kravet til vores system, at støjen på intet tidspunkt må have en større amplitude end selve nyttesignalet.

Vores egen DFT

Vores egen DFT anvender to for-løkker til at bestemme DFT af et givent input-signal inden for et frekvensområde. Vi har genbrugt termene fstart og fend fra FSKgenerator-funktionen. Derudover sendes en samplefrekvens, fs, med som parameter til funktionen.

Vi har implementeret en standard DFT, og ikke en FFT, så den er en smule langsommere end den indbyggede MATLAB funktion. Mere om det senere.


```

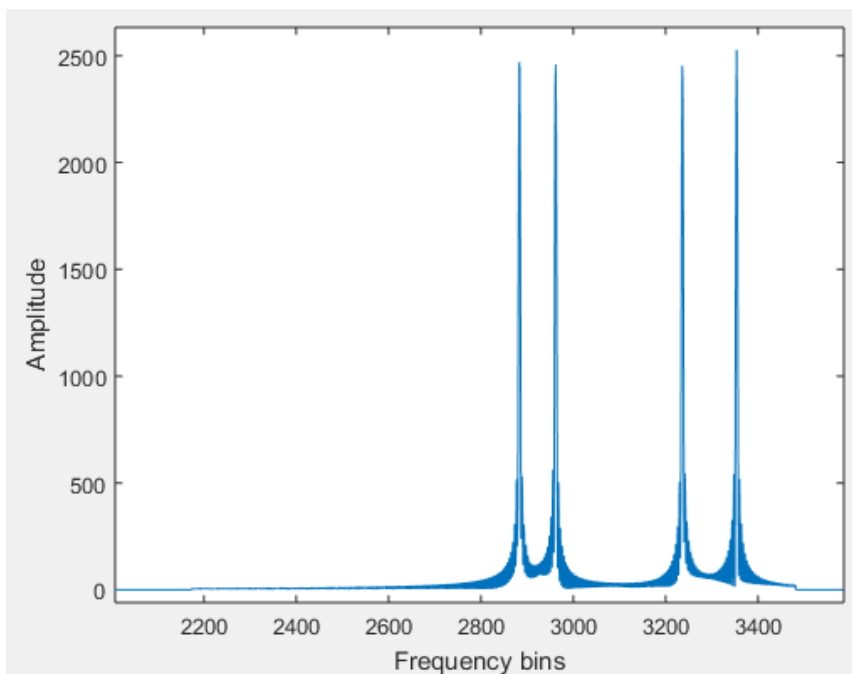
24 function z = Dekodning(x,fstart,fend,fs)
25
26     N = length(x);
27
28     ASCII60 = (60*(fend-fstart)/2^8)+fstart;    %Frequency for ASCII char 60
29     ASCII127 = (127*(fend-fstart)/2^8)+fstart;  %Frequency for ASCII char 127
30
31     f60=ASCII60*N/fs;    %Bin no. for ASCII char 60
32     f127=ASCII127*N/fs; %Bin no. for ASCII char 127
33
34     z = zeros(1, N);    %Array full of zeros
35
36     sig = 0;
37     for k = round([f60:f127 N-f127:N-f60])
38         for n = 1:N
39             sig = sig + x(n)*exp((-2*pi*1i*k*n)/N); %From definition.
40         end
41         z(k+1) = sig;    %Makes the bin number match with the FFT.
42         sig = 0;
43     end
44     return

```

Figur 10 kodeudsnit af vores egen DFT

Vi har optimeret funktionen, ved kun at beregne amplituderne for de frekvenser der ligger inden for intervallet, hvori ASCII karaktererne for A-z er repræsenteret.



Det vises på nedenstående plot, hvor amplituden er 0 uden for ASCII-intervallet, som starter lidt under frequency bin 2200 og slutter omkring bin 3440.



Figur 11 Optimeret DFT

Vores egen implementerede DFT tager en smule længere end MATLABs indbyggede FFT. Vi finder ud af, hvor lang tid det tager at lave DFT med vores egen funktion ved at time en kørsel af scriptet:

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
13	<code>sig = sig + x(n)*exp((-2*pi*li...</code>	52360000	13.598 s	64.4%	
14	<code>end</code>	52360000	7.501 s	35.5%	
9	<code>sig = 0;</code>	1	0.010 s	0.0%	
11	<code>if ((k >= ASCII60) &&am...</code>	20000	0.004 s	0.0%	
18	<code>end</code>	20000	0.002 s	0.0%	
All other lines			0.004 s	0.0%	
Totals			21.119 s	100%	

Figur 12 tiden det tager at beregne DFT

Det ses af ovenstående figur at det er linjen hvor selve DFT'en udregnes, der er kritisk.

Opgave 3 – Signal-støj-forhold

SNR beregnes som $10 \cdot \log(P_S / P_N)$.

Det er derfor forholdet mellem effekten af signalet og effekten af støjen.

P_S er summen af af værdierne i intervallet ASCII60 -> ASCII127. Hvor ASCII-værdierne repræsenterer bin numre.

Tsymbol (s)	SNR (dB)	Støjgulv (dB)	Max amplitude (dB)
1	21.659	-74.356	117.124
0.5	16.090	-43.254	140.914
0.1	20.624	-56.594	101.348
0.05	36.970	-36.001	98.837
0.01	20.185	-29.227	77.838

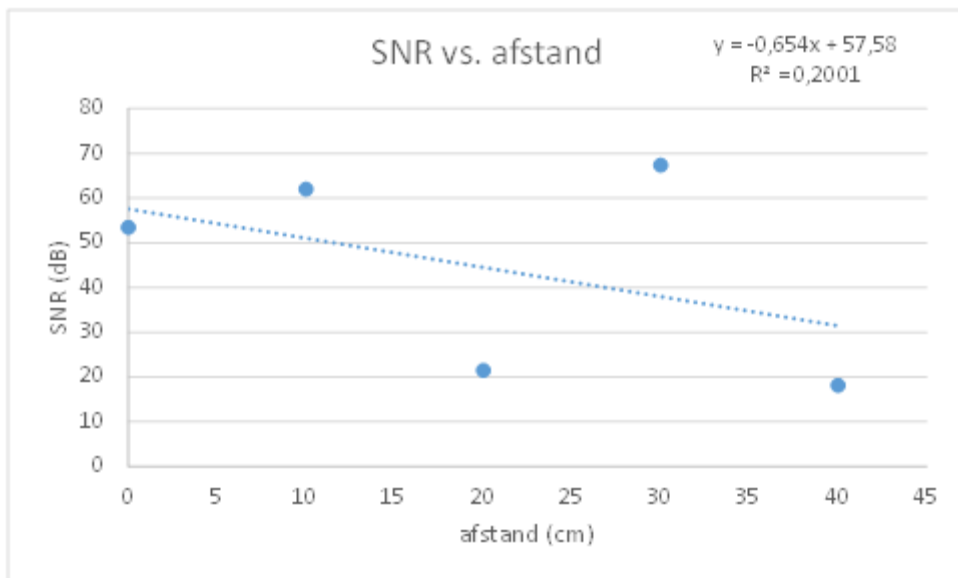
Vi fik meget ulineære målinger, som vi mener skyldes selve optagelsen med mikrofonen.

Vi mener, at mikrofonen har indbygget filtrering og måske automatisk justering i forhold til optagelse af lyd. Det kan altså være at lydkortet selv ændrer forstærkningen af signalet og filtrerer derefter.

Vi forventer at støjgulvet bliver betydeligt højere med afstanden mens max amplitude af signalet falder, når vi mindsker værdien af Tsymbol.

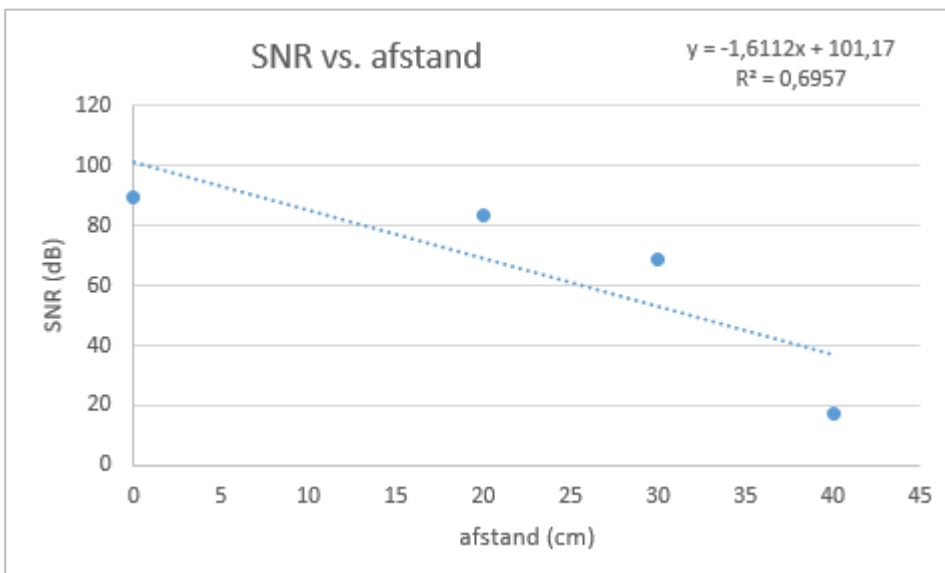
Forholdet mellem støjen og signalet bliver sværere at skille fra hinanden idet vi samler over kortere tid. Dette påvirker SNR betydeligt.

Vi plotter SNR som funktion af afstanden mellem mikrofon og sender.



Figur 13 SNR vs afstand for hele signalet

For at forbedre dette, har vi prøvet at plote et enkelt tone-segment og bestemme SNR som funktion af afstand for denne værdi.



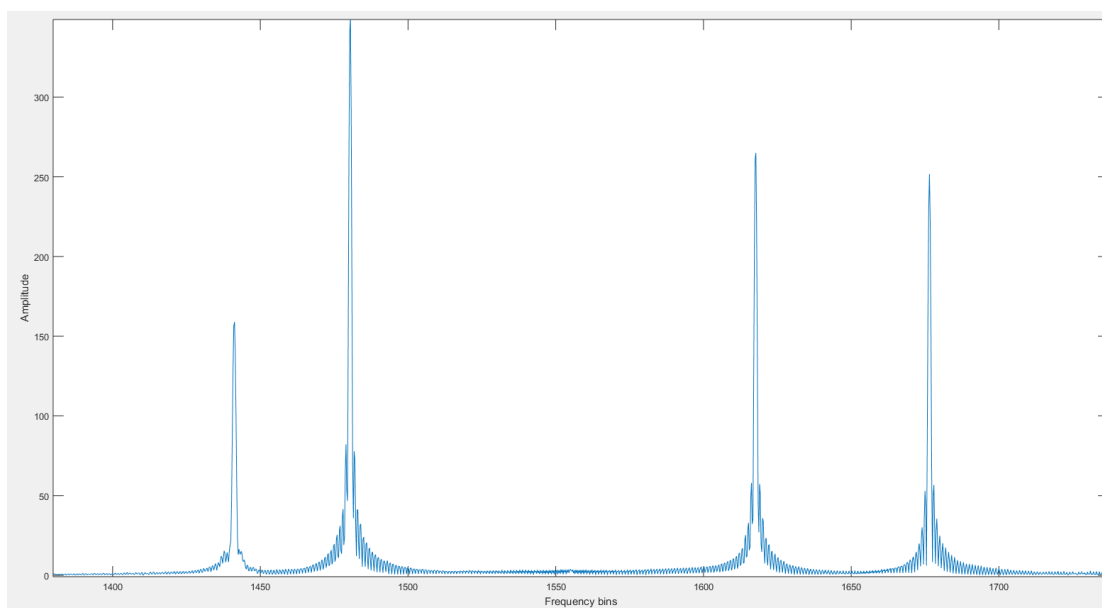
Figur 14 SNR vs afstand er kun for et enkelt tone-segment

Det viser sig at sammenhængen bliver nærmere lineær, når vi plotter et enkelt tone-segment.

Opgave 4 – Bit rate

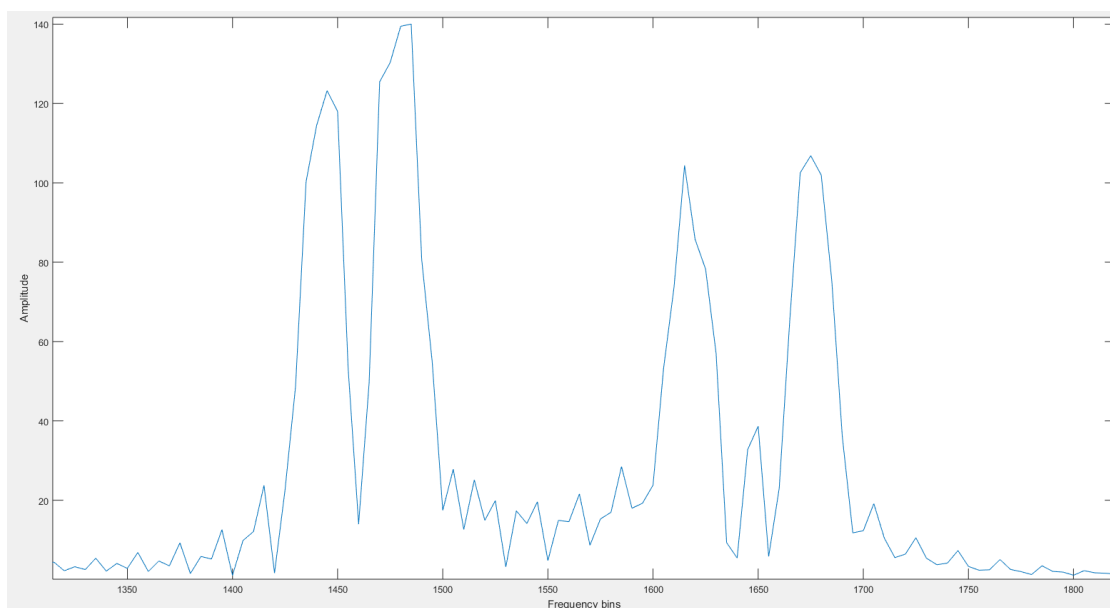
Ændre på Tsymbol

Ved at ændre på Tsymbol sendes hvert bogstav i længere eller kortere tid, og ved at sætte Tsymbol ned sendes den samme sekvens af bogstaver hurtigere og bit raten øges.



Figur 15 fft af signal med Tsymbol på 1 s

I Figur 15 ses en symbol rate på 1 symbol per sekund, hvor signalet er nemt at dekode.

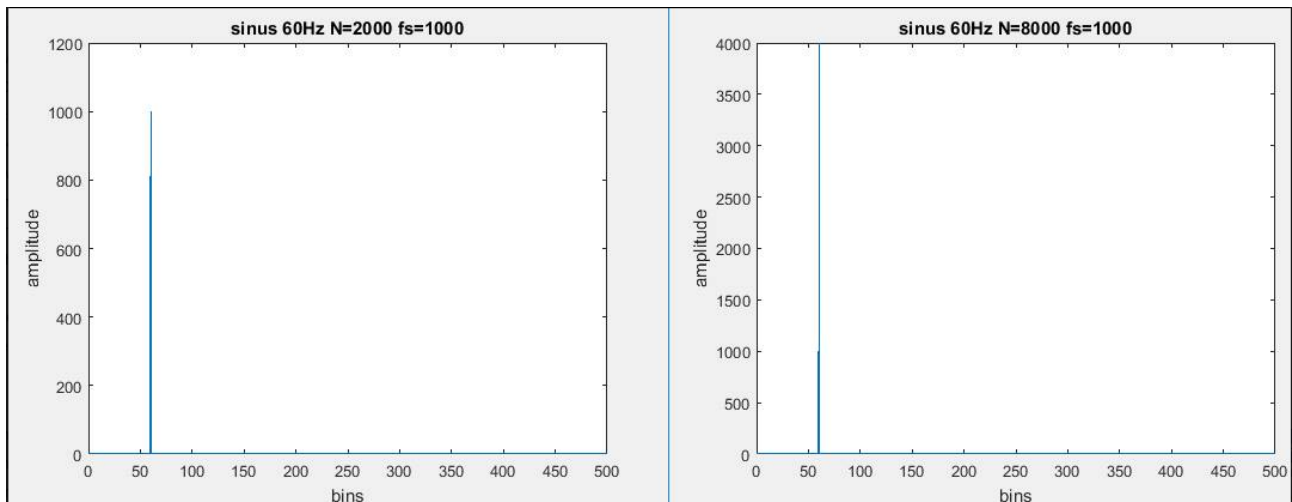


Figur 16 fft af signal med Tsymbol på 0.05 s

I Figur 16 ses den højeste symbol rate vi opnåede på 0.05 symbol per sekund. Ved højere bit rater fås et forvrænget signal som ikke giver den rigtige kombination af bogstaver.

Hvad er betydningen af vindueslængden for amplitude-spektret af en sinus-tone?

Størrelsen af amplituden af signalet er proportionalt med længden af signalet. Det betyder f.eks. at har vi et signal med en varighed på 2 sekund, vil størrelsen af amplituden være 4 gange så stor hvis altså vi i stedet kigger på signalet i 8 sekunder i stedet for 2 sekund. Dette er illustreret i Figur 17. Her ses at amplituden er ligger 4 gange så stort ved signalet på 8 sekunder i forhold til signalet på 2 sekunder.



Figur 17 Venstre: $N=2000$, $T_s=2$ s, Højre: $N=8000$, $T_s=8$ s

Sende med forskellige frekvenser samtidig

Hvis det teoretisk set er muligt at sende og sample med uendeligt stor samplingsfrekvens fra en højttalere og en mikrofon, kan vi i teorien have uendeligt mange forskellige frekvenser inden for samme tid, da en større f_s vil give en større frekvensopløsning. Dette kan ses ved nedenstående formel:

$$\text{frekvens opløsning} = \frac{f_s}{N}$$

Hvor N er givet ved $f_s \cdot T_{\text{symbol}}$.

Det betyder at antal forskellige frekvenser vi kan sende med er begrænset af samplingsfrekvensen og hvor langt T_{symbol} er.

Tradeoffs

Symbol raten er givet ved det inverse af tiden der sendes i. Altså:

$$symbol\ rate = \frac{1}{T_{symbol}}$$

Det betyder at jo større symbol rate vi vil opnå, jo mindre bliver T_{symbol} , altså længden af signalet. Det betyder så, at symbol raten vokser, forværres frekvensopløsningen og det bliver svære at adskille forskellige frekvenser fra hinanden. Dette skyldes SNR.

SNR forværres når vi samler over kortere tid, fordi frekvensopløsningen bliver dårligere. Dette skyldes, at en dårligere frekvensopløsning resultere i større spektral forbreddning, som fordeler energien i signalet i de bins som ligger ved siden af. Dette ses som støj i signalet og i sidste ende kan signal og støj ikke skelnes fra hinanden.